VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Sběr dat za sensorů v mobilním telefonu a následný přenos dat do počítače s OS Linux

Sensor Data Collection from Mobile and Data Transmission to Linux Computer

2020 Bc. Laura Kubicová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Laura Kubicová**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Sběr dat za sensorů v mobilním telefonu a následný přenos dat do počítače s OS Linux
Sensor Data Collection from Mobile and Data Transmission to Linux Computer

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je zajistit pravidelný sběr a zpracování dat ze sensorů mobilního telefonu. Následně budou data přenášena do počítače, kde budou vykreslována pomocí vhodných grafů.

Řešení práce spočívá ve splnění následujících úkolů:
1. Popis problematiky sběru a zpracování dat ze sensorů mobilního telefonu.
2. Popis problematiky vykreslování časově závislých grafů v OS Linux.
3. Návrh řešení automatizovaného sběru a přenosu dat na platformě Android.
4. Zpracovaná data budou vykreslovány pomocí nástroje Cacti.
5. Ověření funkčnosti navrženého řešení a následné zhodnocení.

Seznam doporučené odborné literatury:

[1] Gajjar,M.J. *Mobile Sensors and Context-Aware Computing* Morgan Kaufmann 2017
[2] Rehg, J.M, Murphy, S.A, Kumar, S. *Mobile Health: Sensors, Analytic Methods, and Applications* Springer 2017

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prehlasujem, že som túto diplomovú prácu vypracovala samostatne. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

V Ostrave dňa: 15.května 2020 ……

Rada by som poďakovala vedúcemu diplomovej práce Ing. Pavlovi Nevludovi za odbornú pomoc a vedenie počas tvorby práce.

Abstrakt

Diplomová práca sa zaoberá zberom dát z vybraných senzorov mobilného telefónu a ich následným vykreslením v monitorovacom toole Cacti. Cieľom práce bolo vytvoriť návrh automatizovaného zberu a prenosu dát na platforme Android. Práca obsahuje teoretické informácie o jednotlivých senzoroch mobilných zariadení a ich následné využitie v IoT. Praktická časť práce sa zameriava na princíp fungovania MQTT protokolu a jeho využití v domácej LAN sieti za použitia berry Pi mikropočítača.

Klíčová slova: MQTT, RRDtool, Cacti, IoT

Abstract

This Diploma thesis analyze data gathering from selected senzors of mobile device and subsequently its delineation in Cacti monitoring tool. Objective is to create automated collection and transfer of data from Android platform. Thesis contains theoretical information of various senzors in mobile devices and their use in Internet of Things. Practical part takes apart principle of functioning MQTT protocol and its use in home LAN network with use of RPI mikrocomputer.

Keywords: MQTT, RRDtool, Cacti, IoT

Obsah

Seznam použitých zkratek a symbolů

MQTT	–	Message Queuing Telemetry Transport
PDA	–	Personal Digital Assistant
LAN	–	Local Area Network
3G	–	Third generation of wireless mobile technology
4G	–	Fourth generation of wireless mobile technology
5G	–	Fifth generation of wireless mobile technology
GPS	–	Global Positioning System
BI	–	Backside Illumination
FHSS	–	Frequency Hopping Spread Spectrum
UID	–	Unique Identifier
IoT	–	Internet of Things
BAN	–	Body Area Network
PAN	–	Personal Area Network
CAN	–	Campus Area Network
MAN	–	Metropolitan Area Network
RAN	–	Radio Access Network
WAN	–	Wide Area Network
IEEE	–	Institute of Electrical and Electronics Engineers
LPWAN	–	Low-Power Wide-Area Network
WBAN	–	Wireless Body Area Network
USB	–	Universal Serial Bus
D2D	–	Device to Device
D2S	–	Device to Server
S2S	–	Server to SerDevicever
DDS	–	Data Distribution Service
CoAP	–	Constrained Application Protocol
XMPP	–	Extensible Messaging and Presence Protocol
AMQP	–	Advanced Message Queuing Protocol
QoS	–	Quality of Service
PUBREC	–	Publish Received
PUBREL	–	Publish Release
PUBCOMP	–	Publish Complete
XML	–	eXtensible Markup Language
JSON	–	JavaScript Object Notation
SUBACK	–	Subscribe acknowledgement
UNSUBACK	–	Unsubscribe acknowledgement

DST	–	Data Source Type
PDP	–	Primary Data Points
SNMP	–	Simple Network Management Protocol
NMS	–	Network management station
OID	–	Identifikátory objektov
MIB	–	Management Information Base

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Mobilné telefóny predstavujú sofistikované výpočtové platformy, ktoré sa stali súčasťou našich životov.

S rastúcim výpočtovým výkonom procesorov, vylepšeniami v oblasti úložiska a s pokrokmi v oblasti operačných systémov z nich stávajú zariadenia, ktoré môžu obsluhovať vedecké aplikácie a rovnako spracúvavať a ukladať väčsie množstvo dát.

Senzory boli zabudované do mobilných telefónov, aby tieto zariadenia dokázali reagovať na zmeny vo fyzickom prostredí človeka a súčasne, aby zvýšili ich vyžiteľnosť. Inteligentné zariadenia vybavené senzormi zbierajú údaje o spôsobe akým sú využívané a to napr. údaje o teplote, vlhkosti, zvuku ale aj informácie o lokalizácií. Negatívnym faktom je, že dáta zo senzorov mobilných zariadení môžu byť získané a zneužité tretími stranami, neoprávnene.

Od novej generácie mobilných telefónov sa očakáva integrácia nových sofistikovanejších typov senzorov, ktoré zvýšia ich "inteligenciu"a zároveň umožnia mnohým aplikačným oblastiam poskytnúť lepšie služby.

Vzájomné prepojenie výpočtových zariadení, ktoré sú schopné prenášať dáta cez sieť poznáme pod pojmom Internet of the Things. Internet of the Things pomáha ľuďom žiť a pracovať "inteligentnejšie", ponúka možnosť prepojenia zariadení za účelom automatizácie domovov či kancelárií. Dob a"Internetu vecí" nám prináša mnohé výhody a zjednodušenia každodenných životov. Rovnako však prináša aj negatíva, ktoré môžu predstavovať napr. získavanie a zneužitie osobných dát inými osobami.

Táto práca sa zaoberá zberom dát zo senzorov mobilného telefónu a ich následným vykreslením v monitorovacom toole Cacti. Cieľom bolo zaistiť, aby sa dáta prenášali a vykresľovali v reálnom čase, na čo bol využitý mikropočítač Raspberry Pi.

V teoretickej časti práce sú rozdelené a opísané štandardné senzory mobilného telefónu. Súčasne sa v rámci práce venujem i MQTT protokolu, ktorý bol nosným protokolom pri spracuvávaní tejto práce.

2 Mobilné telefóny a senzory

2.1 Všeobecné rozdelenie mobilných telefónov a senzorov

Mobilné zariadenie predstavuje viacúčelovú výpočtovú platformu, ktorú môžeme definovať na základe jej funkcií.

- Mobilné telefóny s nižším výkonom (Low-end devices) poskytujú základné funkcie ako telefonovanie, odosielanie správ, spravovnie kontaktov. Disponujú staršou verziou operačného systému, 2G maximálne 3G sieť. Príznačná je malá až stredná veľkosť obrazovky, malá pamäť telefónu a nižšia kvalita fotoaparátu. [1]
- Mobilné telefóny s vyšším výkonom (High-end devices) prichádzajú s najnovšou verziou operačného systému, pričom zariadenia umožňujú uskutočňovať telefonáty a súčasne byť v režime PDA. [1]
  Inteligentné telefóny (Smartphones) umožňujú užívateľom byť v režime spracovania viacerých úloh(multitasking) a to nezávisle na operačnom systéme. Prichádzajú s najnovšími štandardami Wi-fi LAN spojení a technológiami 3G, 4G a v blízkej budúcnosti i 5G.

Mobilný telefón pozostáva z viacerých senzorov, ktoré zvyšujú jeho použiteľnosť a ovládateľnosť.

Senzor je zariadenie, ktoré je schopné zachytávať a reagovať na zmeny vo fyzickom prostredí a následne tieto zmeny v podobe dát odoslať operačnému systému alebo procesoru. Dáta sú schopné zaznamenať, kategorizovať a preniesť s vysokou presnosťou.

Užitočnými sa stávajú pri snímaní polohy telefónu v trojrozmernom priestore, pri detekovaní blízkych objektov v jeho okolí či detekovaní svetla v jeho okolí.

Senzory s ohľadom na snímanie rozdeľujeme do dvoch kategórií.

- Interné senzory sú integrované v mobilnom telefóne pričom vyhodnocujú jeho stav.
- Externé senzory predstavujú senzory, ktoré môžeme k telefónu pripojiť, prípadne s ním môžu komunikovať pomocou bezdrôtových technológií. [1]

Senzory mobilných telefónov delíme do troch hlavných kategórií v závislosti od ich funkcií, využitia a zloženia.

Rozdelenie senzorov na základe funkcionality.

- Aktívna funkcionalita znamená, že dáta získané zo senzora sa využívajú tak ako boli navrhnuté vývojármi.
- Pasívna funkcionalita znamená, že dáta získané zo senzora boli interpretované.

Rozdelenie senzorov v závislosti na prostredie.

- Senzory pohybu merajú akcelaračné a rotačné sily pozdĺž troch osí. Príkladom takýchto senzorov je akcelerometer, gravitačný senzor, gyroskop či rotačný senzor.
- Senzory polohy merajú fyzickú polohu zariadenia. Medzi takéto senzory patrí magnetometer a senzory orientácie.
- Senzory, ktoré merajú parametre prostredia akými sú teplota, tlak, osvetlenie či vlhkosť.

Senzory môžeme tiež rozdeliť na fyzické a virtuálne.

- Fyzické senzory sú založené na hardware(hardware-based) a sú priamo integrované do mobilného telefónu či inteligentého zariadenia. Tieto senzory získavajú dáta priamym meraním vlastností prostredia. Medzi takéto senzory patrí akcelerometer, gyroskop či proximity senzor.
- Virtuálne senzory (kompozitné senzory) sú založené na software (software-based). Tieto senzory získavajú svoje dáta z jedného alebo viacerých hardwarových senzorov. Príkladom virtuálného senzoru je senzor lineárneho zrýchlenia, senzor gravitácie alebo senzor priblíženia.

2.2 Interné senzory

Interné senzory sú integrované senzory, ktoré sú poskytované ako súčasť mobilných zariadní.

Senzor priblíženia

Je senzor, ktorý detekuje prítomnosť objektov bez fyzického kontaktu. Senzor priblíženia uvoľní elektromagnetické žiarenie a tak detekuje prítomnosť cudzieho objektu v blízkosti. Snímaný objekt je označovaný ako cieľ snímača priblíženia.
V problematike mobilných zariadení senzor priblíženia zisťuje ako blízko je tvár užívateľa k obrazovke zariadenia. Príkladom z praxe môže byť situácia, keď sa obrazovka zariadenia vypne a zamkne ako prevencia pred náhodným stlačením tlačidla a šetrenia batérie. Akonáhle je zariadenie odobrané z tesnej blízkosti tváre užívateľa, je uvedené do pôvodného stavu.

Senzor okolitého svetla

Senzor sníma okolité svetlo, ktoré vhodne upravuje a na základe toho optimalizuje jas mobilného telefónu tak, aby bola viditeľnosť na obrazovke v zhode s okolím. [2]Takto zabraňuje, aby bola obrazovka príliš svetlá v tmavej miestnosti alebo príliš svetlá v pristore s dostatkom svetla. Tlmenie svetla na mobilnom zariadení tiež predlžuje životnosť batérie.

Akcelerometer

Akcelerometer alebo tiež senzor zrýchlenia je používaný ako ovládač užívateľského rozhrania. Sníma zrýchlenie v troch osiach *X,Y,Z* na detekciu orientácie a následnú úpravu rozhrania a v závislosť od držania zariadenia. [2] Akcelerometre môžu byť použité na rozpoznávanie aktivít užívateľov.

Senzor vlhkosti

Senzor vlhkosti meria relatívnu vlhkosť prítomnú v prostredí. Senzor vlkosti využíva kapacitné meranie. Využite môže nájsť pri určovaní príčin poškodenia mobilného telefónu, či bolo zariadenie poškodené pôsobením vody alebo nie.

Digitálny kompasový senzor

Digitálny kompasový senzor alebo tiež magnetometer je používaný na rozpoznávanie severu, určuje polohu užívateľa v priestore. Meria magnetické pole pozdĺž osí *X, Y* a Z. [2] Magnetometre používané v mobilných zariadeniach využívajú na fungovanie *Hallov efekt*.

Barometrový senzor tlaku

Je nízkonapäťový, nízkotlakový senzor s vysokým rozlíšením používaný meterológmi na meranie atmosferického tlaku, ktorý sa efektívne používa na predpoveď počasia. V praxi sa barometer používa na zlepšenie výsledkov GPS. V porovnaní s barometrom, GPS pracuje pomalšie a menej presne, pretože barometer pri počítaní pracuje s jednou z troch hodnôt (t.j. zemepisná šírka, zemepisná dĺžka a nadmorská výška).

Gyroskopický senzor

Gyroskopický senzor je senzor pohybu a rovnako ako akcelerometer. Využíva sa na kalkuláciu a výpočet osi otáčania, sníma uhlovú rotačnú rýchlosť a zrýchlenie.
V kombinácii s akcelerometrom umožní zariadeniu merať pohyb po šiestich osiach a presnejšie možnosti meranie pohybu skol zariadenia.

GPS senzor

GPS je navigačný systém, kde GPS prijímače dostavajú informácie poslané GPS satelitmi a vypočítavajú presné umiestnenie užívateľa pomocou triangulácie.

Senzor podsvietenia

Rovnako známy ako senzor osvetlenia zadnej strany (BI) je digitálny zobrazovací senzor, ktorý vylepšuje výkon pri slabom osvetlení a zvyšuje množstvo svetla počas snímania a tým zvýrazňuje zobrazovacie prvky.

Senzor mikrofónu

Je elektromechanický senzor, ktorý detekuje tlak vzduchu a vytvára elektrický signál, ktorý je úmerný vibrácií. V mobilných telefónoch slúžia mikrofónové senzory na nahrávanie hlasu či dopravy.

Bluetooth senzor

Je nízkoenergetický radiokomunikačný senzor navrhnutý predovšetkým na prepojenie periférií. Pracuje v ISM pásme 2,4 GHz a na prenos používa FHSS medzi 79 frekvenciami.

2.3 Externé senzory

Okrem senzorov, ktoré sú integrované priamo v mobilnom telefóne, môže zariadenie komunikovať so senzormi externe či už pomocou technológií krátkeho dosahu (Bluetooth, Wi-fi) alebo prenosom informácií cez dátové káble. Externé senzory môžu poskytovať významné dáta, ktoré môžu byť následne spracované v rôznych odvetviach.

Senzor teploty

Poskytuje informácie o okolitej teplote, pričom rozoznávame dva typy senzorov teploty, kontaktné a nekontaktné.

- Kontaktné senzory poskytujú informácie o teplote objektu, s ktorými sú fyzicky spojené.
- Nekontatné senzory poskytujú informácie o teplote objektu, s ktorými nie sú fyzicky spojené. Senzory teploty našli využitie v produktoch každodenného užitia termostaty, radiátory, či mikrovlnné rúry. [2]

Senzor vlhkosti

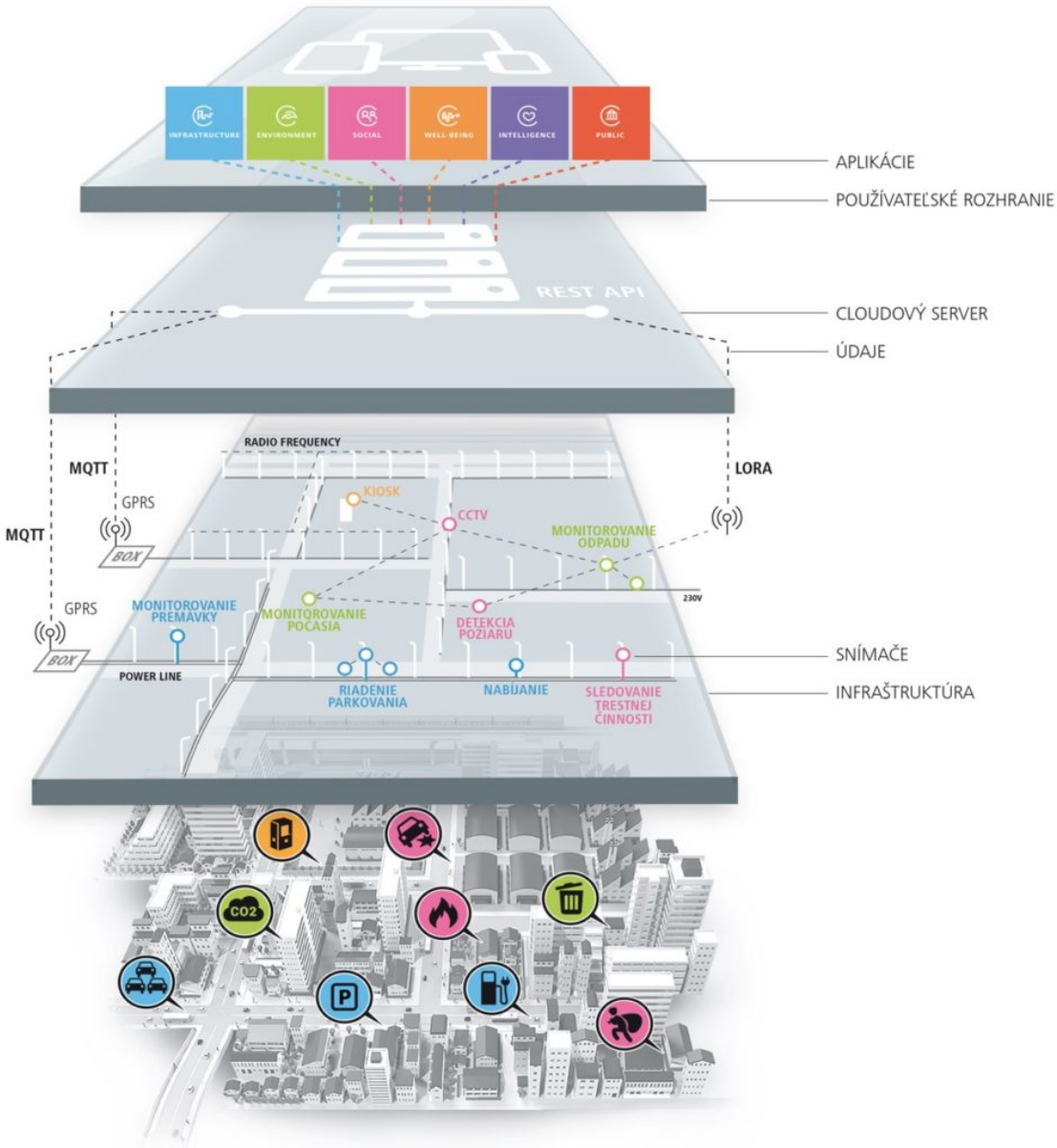
Senzor vlhosti (hydrometer) meria relatívnu vlhkosť v prostredí, s použitím kapacitného meranie. Tento senzor nájde využitie v priemysle, zdravotníctve, enviromentálnom riadení.

2.4 IoT

Internet of things ("Internet vecí") zjednodušene povedané označuje fyzické zariadenia na celom svete, ktoré sú pripojené k internetu, a ktoré zhromažďujú a zdieľajú údaje. Zariadenia pozostávajú z jedinečných identifikátorov (UID), ktoré sú zariadeniam pridelené za účelom identifikácie.

Veci (things)predstavujú objekty, ktorým sú pridelené IP adresy, a ktoré sú schopné prenášať dáta cez sieť bez interakcie človeka.
Zariadenia, ktoré spoločne formujú "Internet vecí" sú vybavené senzormi, ktoré zbierajú údaje o spôsobe akým sa zariadenia využívajú. Takýmito dátami môžu byť napr. údaje o teplote, vlkosti,

zvuku, alebo informácie o geografickej polohe užívateľa.
Na úrovni podniku integrácia IoT môže priniesť výhody v oblasti zprehľadnenia fungovania systémov v reálnom čase, súčasne poskytuje možnosť automatizovať procesy a prípadne znižovať náklady na pracovnú silu.
[3] IoT tvorí troj-úrovňová architektúra pozostávajúca zo: *zariadenia*, *brány* a *data systémov*.



Obrázek 1: Architektúra IoT.

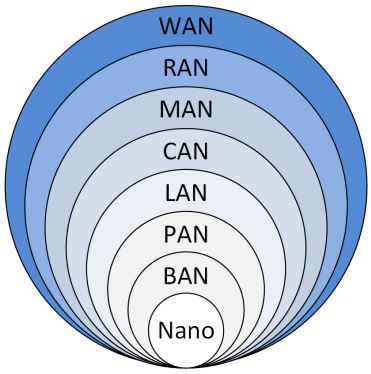
Výhody využitia IoT:

- Automatizácia umožňuje vylepšiť kvalitu servisu a kontrolu denných úloh bez zásahu človeka.
- Efektívnosť predstavuje interakciu medzi jednotlivými zariadeniami, ktorá pomáha udržiavať určitú úroveň transparentnosti v procesoch.
- IoT významne šetrí náklady a výdavky podnikateľom, prípadne znižuje problémy s poruchami či poškodením systémov.
- Umožňuje okamžitý prístup k dátam, tie je možné ukladať v utajenom formáte kvôli bezpečnosti a v prípade, že k nim budeme chcieť pristupovať neskôr.

Neýhody využitia IoT:

- Ochrana osobných údajov a bezpečnosť je negatívom, pretože zariadenia a služby, ktoré zbierajú informácie o nás sa pripájajú na Internet, čo môžu zneužiť neautorizovaní užívatelia.
- Kompatibilita je v súčasnosti problémom, pretože neexistujú medzinárodné štandardy zariadení IoT, čo sťažuje zájomnú komunikáciu zariadení od rôznych výrobcov.
- Komplexnosť a to v prípade, že zlyhá jedno zariadenie alebo hardware, čo môže mať obrovské následky na celú sieť.
- V personálnej oblasti prináša menej pracovných miest, pretože s automatizáciou klesá požiadavka nekvalifikovanej pracovnej sily. Toto môže viesť k určitej miere nezamestnanosti v nekvalifikovanom odvetví. [3]

2.5 IOT siete a ich pokrytie



Obrázek 2: Typy sietí a ich pokrytie.

Nano

Skupina malých zariadení vo vzdialenosti niekoľko nanometrov alebo niekoľko mikrometrov. Týmito zariadeniami sú senzory, ovládače, výpočtové zariadenia. [4]

Body Area Network

Bezdrôtová sieť pre zariadenia v blízkosti alebo vo vnútri tela. Zariadenia zahŕňajú nositeľné technológie ako sú inteligentné hodinky, zariadenia vo vnútri tela ako implantáty.
Tento typ siete je známy ako WBAN a najnovší štandard je IEEE 802.15.6.

Personal Area Network

Spája zariadenia v pracovnom priestore osoby. Siete PAN môžu byť bezdrôtové ale i káblové (používajú napr. USB konektor).
Bezdrôtová PAN je technológia na krátku vzdialenosť, medzi ktoré patrí Bluetooth-IEEE 802.15.1 alebo Zigbee - IEEE 802.15.4.

Local Area Network

Označenie siete na geograficky limitovanom území, najznámejšie technológie Ethernet - IEEE 802.3 a Wi-Fi -IEEE 802.11.

Campus Area Network

Prepojenie viacerých LAN sietí do dosahu 1km až 5 km.

Metropolitan Area Network

MAN je sieť, ktorá spája viac LAN sietí, ktoré môžu byť prepojené do WAN siete. Typickou technológiou, ktorá sa tu používa je MetroEthernet.

Radio Access Network

RAN používa technológiu rádiového prístupu. Technológia RAN sa používa od začiatku bunkovej technológie. Zahŕňa 3G, 4G a 5G siete.

Wide Area Network

Rozsiahla celosvetová sieť, ktorá prepája LAN siete vzdialené niekoľko kilometrov. LPWAN je sieť navrhnutá na komunikáciu na veľké vzdialenosti pri nízkej bitovej rýchlosti. LPWAN sieť môžeme použiť na vytvorenie bezdrôtovej siete senzorov.

Typy komunikácie a protokoly

- **Device to device (D2D)** - Komunikácia prebieha medzi dvoma zariadeniami. Protokol: DDS.
- **Device to server (D2S)** - Komunikácia zariadenia so serverom. Protokoly: MQTT, CoAP, XMPP.
- **Server to sever (S2S)** - Komunikácia dvoch zariadení. Protokol: AMQP.

2.6 MQTT protokol

MQTT je jednoduchý protokol na odosielanie správ pre malé senzory a mobilné zariadenia, optimalizovaný pre siete s vysokou latenciou alebo nespoľahlivé siete. Protokol MQTT poskytuje škálovateľný a nákladovo efektívny spôsob pripojenia zariadení cez internet. [6] Bol vymyslený v roku 1999 inžiniermi Andy Stanford-Clarkom (IBM) a Arlen Nipperom(Arcom, v súčasnosti Cirrus Link).
Pričom špecifikovali niekoľko požiadaviek pre budúci protokol:

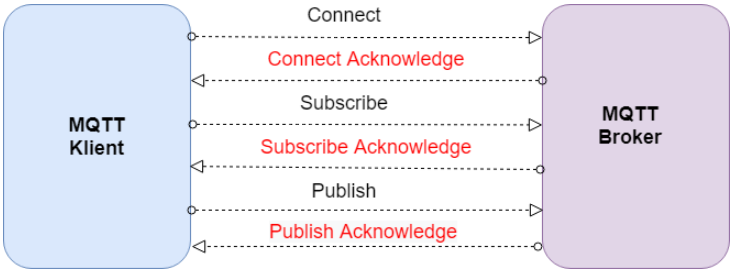
- Jednoduchá implementácia
- Kvalita poskytovania údajov o službách
- Ľahká a efektívna šírka pásma
- Dáta agnostické
- Nepretržité povedomie o relácii

Protokol je postavený na báze *publish/subscribe* a protokole TCP/IP a definuje dva typy sieťových entít: *broker* a *klient*.

Broker je v strede každého publish/subscribe modelu. Je zodpovedný za prijatie všetkých správ, ich filtrovanie a následne posielanie na klientov, ktorí sa o odber prihlásili *subscriberi*. Uchováva relácie (sessions) všetkých pretrvávajúcich klientov a zároveň ich autentizuje a autorizuje. Broker je centrálnym bodom, ktorým prechádza každá správa, preto je nutné aby bol škálovateľný a odolný voči zlyhaniu.
[6]

Klient v terminológií MQTT predstavuje MQTT klienta. Obaja, *publisher* a *subscriber*, sú klientami. MQTT klient môže byť akékoľvek zariadenie od mikroprocesora po server, ktoré spravuje MQTT knižnicu a pripája sa na brokera vrámci jednej siete.
Jednoduchá implementácia je dôvodom, prečo je MQTT tak obľúbené a používané na malých zariadeniach. MQTT knižnice sú dostupné pre veľké množstvo programovacích jazykov ako napr. Arduino, C, C ++, Go, iOS, Java, JavaScript a .NET.

[7]
MQTT spojenie existuje vždy medzi klientom a brokerom, klienti sa nikdy navzájmne neprepájajú. Na inicializáciu spojenia odošle klient CONNECT správu (message) na brokera, ktorý odpovedá CONNACK správou a statusom správy. Akonáhle je spojenie vytvorené, broker ponecháva otvorený port, pokiaľ klient neodošle DISCONNECT správu až pokiaľ sa spojenie nepreruší.
Ak bola správa od klienta chybná alebo by došlo príliš veľa času medzi otvorením socketu a odoslaním CONNECT správy - broker ukončí spojenie. Toto správanie odráža škodlivých klientov, ktorí brokera spomaľujú. [8]



Obrázek 3: MQTT spojenie.

2.7 Formát kontrolného paketu

MQTT protokol funguje definovanou výmenou série kontrolnych paketov, ktoré nesú dáta.

Štruktúra kontrolného paketu

Pevná MQTT hlavička, prítomná vo všetkých kontrolných paketoch
Premenná MQTT hlavička, prítomná v niektorých kontrolných paketoch
Zaťaženie, prítomné v niektorých kontrolných paketoch MQTT

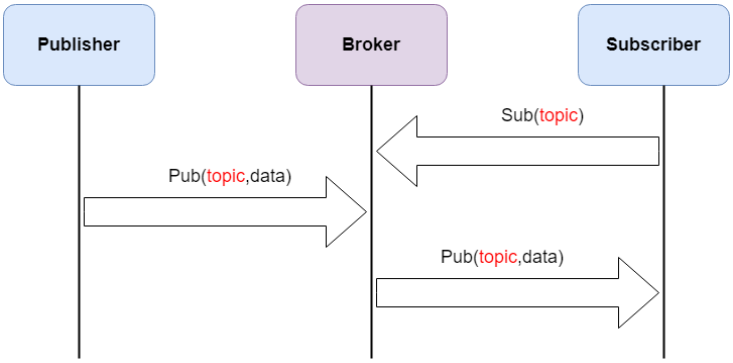
Tabulka 1: Štruktúra kontrolného paketu MQTT.

Meno	**Hodnota**	**Smer toku**	**Popis**
Rezervovaný	0	Zakázaný	Rezervovaný
CONNECT	1	Klient k serveru	Požiadavka klienta k pripojeniu
CONNACK	2	Klient k serveru	Potvrdenie pripojenia
PUBLISH	3	Klient k serveru/Server ku klientovi	Publish správa
PUBACK	4	Klient k serveru/Server ku klientovi	Publish potvrdenie
PUBREC	5	Klient k serveru/Server ku klientovi	Publish správa prijatá
PUBREL	6	Klient k serveru/Server ku klientovi	Uvoľnenie správy publish
PUBCOMP	7	Klient k serveru/Server ku klientovi	Publish správa je kompletná
SUBSCRIBE	8	Klient k serveru	Klient subscribe žiadosť
SUBACK	9	Server ku klientovi	Subscribe potvrdenie
UNSUBSCRIBE	10	Klient ku serveru	Unsubscribe žiadosť
UNSUBACK	11	Server ku klientovi	Unsubscribe potvrdenie
PINGREQ	12	Klient ku serveru	PING žiadosť
PINGRESP	13	Server ku klientovi	PING odpoveď
DISCONNECT	14	Klient ku serveru	Client sa odpája
Reserved	15	Zakázaný	Rezervovaný

Tabulka 2: Kontrolné typy paketov.

Publish/Subscribe komunikácia

Poskytuje alternatívu k tradičnej architektúre *klient-server*, v ktorom klient komunikuje priamo s koncovým bodom - severom. V modeli publish/subscribe, je klient - odosielateľ správ - oddelený od prijímateľov správy. Klient, ktorý odosiela správu je - publisher klient, ktorý správu prijíma je subscriber.
Publisher a subscriber sa priamo nekontaktujú, spojenie vykonáva tretia strana a to broker, ktorého úlohou je sprostredkovať a filtrovať všetky prichádzajúce správy a distrubovať ich subscriberovi.
[9]



Obrázek 4: MQTT komunikácia.

Najdôležitejším aspektom publish/subscribe komunikácie je oddelenie publishera a subscribera. Toto oddelenie má niekoľko rozmerov.

- *Oddelenie podľa miesta* - Publisher a Subscriber nemusia o sebe vedieť, napríklad si nemenia IP adresu či číslo portu.
- *Časové oddelenie* - Publisher a Subscriber nepotrebujú fungovať v rovnakom čase.
- *Oddelenie podľa miesta* - Počas publikovania a prijímania správ nie je nutné prerušovať operácie na oboch komponentoch.

Zjednodušene povedané, publish/subscribe metóda odstráni priamu komunikáciu medzi vykonávateľom/vydávateľom správy a prijímateľom správy. Táto filtrovacia funkcia brokera umožňuje rozhodnúť, ktorý subscriber príjme správy.
Ďalšou významnou funkciou je *škálovateľnosť*, ktorá v modeli publish/subscribe škáluje lepšie ako v modeli klient-server. Je to spôsobené tým, že operácie na brokerovi môžu byt paralyzované a správy môžu byť spracovávané riadeným spôsobom (event-driven way).

Filtrovanie správ

Broker hrá ústrednú úlohu v publish/subsribe procese tak aby by každý účastník dostával správy, o ktoré má záujem. Umožňuje nám to proces filtrovania správ (message filtering).

1. Filtrovanie správ podľa témy je založené na téme/predmete správy. Prijímajúci klient sa prihlási k odberu správ, o ktoré ma záujem u brokera. Od toho momentu broker zabezpečí, aby klient dostával všetky publikované správy, o ktoré mal záujem.
   Vo všeobecnosti,sú témy textové reťazce (stringy) v hierarchickej štruktúre, ktoré môžeme filtrovať na základe obmedzeného počtu výrazov.

2. Filtrovanie správ podľa obsahu je typ filtrovania, kde broker filtruje správy na základe špecifickeho “obsahového” jazyka. Významnou nevýhodou tohto filtrovania je fakt, že obsah správ musí byt vopred známy a nedá sa šifrovať alebo meniť.

3. Filtrovanie správ podľa typu, kde subscriber môže odoberať všetky správy, ktoré sú typu *Exception* alebo iného sub-typu. [9]

Oddelenie publishera/subscribera je kľúčovou úlohou, ale prináša aj niekoľko výziev ako napr. musíte vopred vedieť ako sú jednotlivé dáta štruktúrované.

V prípade filtrovania správ podla témy musia obaja - publisher a subscriber - vedieť akú tému budú používať. Ďalšou vecou, ktorú treba mať na pamäti je, že publisher nemôže predpokladať, že jeho správy niekto odoberá, pretože v určitých prípadoch konkrétnu tému nikto neodoberá.

Kvalita služieb

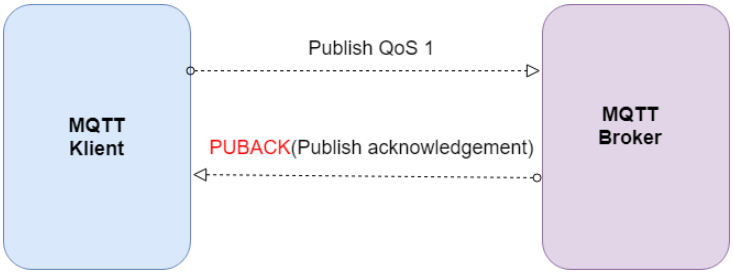
QoS úroveň kvality služby je "dohoda" medzi odosielateľom a príjemcom správy, ktorá definuje záruku doručenia konktrétnej správy. QoS je kľúčovou vlastnosťou MQTT protokolu, pretože umožňuje klientovi zvoliť si takú úroveň služieb, ktorá zodpovedá jeho spoľahlivosti siete a aplikačnej logike. QoS uľahčuje komunikáciu v nespoľahlivých sieťach. Keď hovoríme o MQTT, musíme zvážiť dve strany doručenia správy. [10]

- Odosielanie správ z publishera na brokera.
- Odosielanie správ z brokera na subscribera.

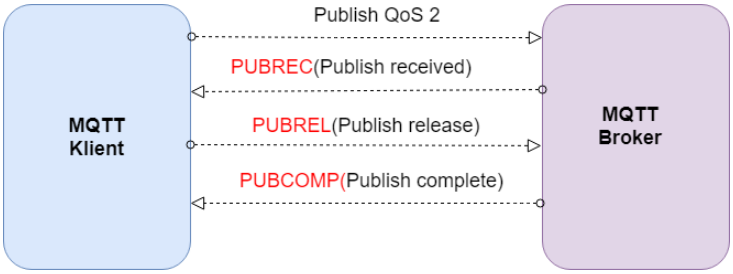
Klient, ktorý odosiela správu na brokera definuje úroveň QoS správy. Broker odošle túto správu pomocou úrovne, ktorú si definoval subscriber pri prihlásení o odber správ. Pokiaľ si subscriber definuje nižšiu QoS ako publisher, tak broker odošle správu s nižším QoS.
MQTT protokol pozná 3 úrovne QoS.

1. QoS 0 je minimálna úroveň, ktorá zaručuje najlepšie (best-effort) doručenie, avšak neexistuje žiadna záruka doručenia. Príjemca nepotvrdí prijatie správy, rovnako správa nie je uložená ani odoslaná ďalej.
2. QoS 1 zaručuje, že spáva je doručená aspoň raz. Publisher ukladá správu pokým nedostane PUBACK paket od brokera. Správa môže byť odoslaná alebo prijatá niekoľkokrát.
3. QoS 2 je najvyššia úroveň, ktorá zaručuje, že každá správa je prijatá brokerom iba raz. Je to najbezbečnejšia a najpomalšia úroveň QoS. Záruka doručenia správ je zabezpečená najmenej dvoma request/response tokmi (four-part handshake) medzi odosielateľom a prijímateľom.

Obrázek 5: Úroveň kvality služieb 0.



Obrázek 6: Úroveň kvality služieb 1.



Obrázek 7: Úroveň kvality služieb 2.

Typy správ

PUBLISH správy môže klient posielať na brokera akonáhle sa na neho pripojí. Každá správa musí obsahovať *topic*, ktorý broker využíva na odosielanie správ k subscriberom. Každá správa musí obsahovať *payload*, ktorý obsahuje údaje v bajtovom formáte. Odosielajúci klient sa rozhodne, či chce dáta posielať binárne, alebo preferuje textové údaje či XML alebo JSON.

Publish správa obsahuje nasledujúce atribúty.

- *Názov témy* je string hierarchicky oddelený lomítkami.
- *QoS* označuje úroveň kvality služby.
- *Ponechávajúca vlajka* príznak určuje, či správa bude brokerom uložená ako posledná dobre známa hodnota pre danú tému. Keď sa nový klient prihlási na odber témy, dostane poslednú správu, ktorá je k tejto téme zachovaná.

- *Payload* predstavuje skutočný obsah správy. MQTT umožňuje posielať správy, text či obrázky v ľubovolnom kódovaní či šifrovaní.
- *Identifikátor paketu* jedinečne definuje správu pri jej toku od klienta po brokera. Identifikátor paketu je relevantný pri QoS väčšej ako 0.
- *DUP vlajka* definuje, že správa je duplikátom a bola opäť odoslaná, pretože príjemca - klient alebo broker - pôvodnú správu nepotvrdil.

Klient, ktorý odosiela publish správy sa zaujíma iba o odoslanie tejto správy. Akonáhle správu prevezme broker je jeho povinnosťou odoslať správy na odoberateľov. Klient, ktorý odosiela publish správu nedostáva žiadne potvrdenie o tom, že sa na odoberanie správy niekto prihlásil, rovnako ako nedostáva potvrednie o tom, koľkí klienti dostali od brokera správu.

Odosielanie publish správ má zmysel práva vtedy, keď niekto tieto správy odoberá. Na prijímanie správa odosiela klient brokerovi správu SUBSCRIBE.

Subscribe správa obsahuje nasledujúce atribúty:

- *Identifikátor paketu* identifikuje správu, ktorá prúdi medzi klientom a brokerom. Knižnica klienta a/alebo broker sú zodpovední za nastavenie tohto identifikátora.
- *Zoznam odberov* subscribe správa môže obsahovať viac odberov témy jedného klienta. Každá odoberaná správa pozostáva z témy a QoS.

SUBACK je správa, ktorú posiela broker klientovi na potvrdenie odberu.

- *Identifikátor paketu* identifikuje správu. Tento kód je rovnaký ako v správe subscribe.
- *Návratový kód* odosiela broker ako kód pre - tému /QoS -, ktorú dostane v správe subscribe. Takže pokiaľ subscribe správa má dva odbery, SUBACK obsahuje dva návratové kódy. Návratový kód potvrdzuje každú správu a ukazuje QoS level, ktorý garantuje broker. Pokiaľ broker odmietne odber, SUBACK správa obsahuje porušený návratový kód.

Návratový kód	Odpoveď na návratový kód
0	Úspech - maximálne QoS 0
1	Úspech - maximálne QoS 1
2	Úspech - maximálne QoS 2
128	Zlyhanie

Tabulka 3: Návratové kódy pre SUBACK správy.

Protiváhou k správe subscribe je správa UNSUBSCRIBE. Táto správa odstraňuje všetky existujúce odbery klienta na brokerovi.

Táto správa je podobná správa subscribe tým, že obsahuje:

- *Identifikátor paketu* identifikuje správu, ktorá prúdi medzi klientom a brokerom. Knižnica klienta a/alebo broker sú zodpovední za nastavenie tohto identifikátora.
- *Zoznam odberov* môže obsahovať niekoľko tém, ktoré chce klient prestať odoberať. Je nutné odoslať len tému(bez QoS). Broker odstráni tému z odberu bez ohľadu na to, pod akým QoS bola pôvodne odoberaná.

UNSUBACK je potvrdzujúca správa zrušenia odberu, ktorú odosiela broker klientovi. Táto správa obsahuje len identifikátor paketu povodnej UNSUBSCIRBE správy pre jasnú identifikáciu správy. [9]

```
Subscribe a publish komunikacia
root@raspberrypi:~# mosquitto_sub -d -t testtopic
Client mosqsub|5474-raspberryp sending CONNECT
Client mosqsub|5474-raspberryp received CONNACK (0)
Client mosqsub|5474-raspberryp sending SUBSCRIBE (Mid: 1, Topic: testtopic, QoS
    : 0)
Client mosqsub|5474-raspberryp received SUBACK


root@raspberrypi:~# mosquitto_pub -d -t testtopic -m "hello world"
Client mosqpub|5499-raspberryp sending CONNECT
Client mosqpub|5499-raspberryp received CONNACK (0)
Client mosqpub|5499-raspberryp sending PUBLISH (d0, q0, r0, m1, 'testtopic',
    ... (11 bytes))
Client mosqpub|5499-raspberryp sending DISCONNECT
```

Výpis 1: Subscribe a publish komunikácia

3 Vytváranie grafov v OS linux

3.1 RRD tool

RRDtool (Round-robin database tool) je open source nástroj na spracovanie a ukladanie časovo závislých dát. RRDtool slúži k montitorovaniu dát a ich následnému vykresľovaniu v reálnom čase.

3.2 Vytvorenie databáze

- Databázu vytvoríme pomocou príkazu `rrdtool create`.
- Opakované pridávanie dát môžeme zaistiť pomocou scriptu a `rrdtool update` príkazu. [14]
- Funkcia `rrdtool graph` slúži na vykreslenie dát do grafov.

```
#Databáza v RRDtool
  rrdtool create filename
  [--start|-b start time]
  [--step|-s step]
  [--no-overwrite]
  [DS:ds-name:DST:dst arguments]
  [RRA:CF:xff:steps:rows]
```

Výpis 2: Ukážka vytvorenia databázy

- `filename` predstavuje názov súboru, ktorý by mal končiť príponou `.rrd`. RRDtool však vezme aj akýkoľvek iný typ súboru.
- `-start|-b start time` určuje odkedy sa nám budú vyresľovať dáta. Tento čas sa odvádza od 1.1.1970 UTC a prednastavená hodnota by mala byť -10 sekúnd.
- `-step|-s step` určuje časový interval, po ktorom budú dáta ukladané do databázy. Predom nastavená hodnota je 300 sekúnd.
- `-no-overwrite` príkaz zaistí, že existujúci súbor s rovnakým názvom neprepíše databázu.

Data Source predstavuje zdroj dát. Do jednej databázy môžeme vkladať viacero hodnôt(premenných), pre ktoré musíme definovať jej základné parametre.

Parametre Data Source

- `DS: ds-name` je názov k odkazu na zdroj dát. Názov má obmedzený počet znakov 19, pričom môže obsahovať len malé a veľké písmená anglickej abecedy, čísla a podtržítko.
- `DST` (Data Source Type) typ zdroja dát. Môže nadobúdať hodntoty: GAUGE, COUNTER, DERIVE, ABSOLUTE alebo COMPUTE.

Data Source Type

- `GAUGE`: typ je určený pre hodnoty dát ako sú využitie pamäte procesora, teplota.
- `COUNTER`: určuje rozdiel medzi aktuálnou a pôvodnou teplotou. Tento typ predpokladá, že sa hodnoty neustále navyšujú.
- `DERIVE`: pracuje podobne ako COUNTER, ale bez kontroly pretečenia. Tento typ môže nadobúdať aj záporných hodnôt.
- `ABSOLUTE`: sa používa pri čítačoch, ktoré sú po každom pretečení resetované.
- `COMPUTE`: slúži pre ukladanie vzorcov aplikovaných na ďalšie dáta v RRD. Tieto dáta neposkytujú hodnoty pre update, ale pre PDPs.
- `Heartbeat`: určuje počet sekúnd, po ktorých je hodnota určená ako neznáma. Slúži k tomu, aby sa nezapisovali hodnoty NaN(not a number).
- `Min a Max`: určuje rozsah hodnôt. Pokiaľ hodnota prekročí daný rozsah, tak sa zapíše ako UNKNOWN. Pokiaľ je nám rozsah neznámy, tak môžeme zadať U pre neznámu veličinu.
- `Rpn-expression`: sa používa iba u COMPUTE a definuje vzorec použitý pre výpočet hodnoty z iných zdrojových dát v rovnakej RRD databáze.

Round Robin Archív

RRA(Round-Robin Archive) definuje ako sa jednotlivé dáta budú ukladať. Pomocou RRA nastavujeme koľko hodnôt na interval sa v databáze uchová. V závislosti na RRA a počte premenných DS je určená veľkosť súboru RRD. [14]

- `CF`: Je typ kosolidačnej funkcie na použité hodnoty - AVERAGE, MAX, MIN, LAST.
- `xxf`: Udáva aká časť konsolidačného intervalu sa môže skladať z neznámych hodnôt, ale výsledná konsolidačná hodnota je považovaná za neznámu.
- `steps`: Určuje, koľko hodnôt sa použije k vytvoreniu konsolidovanej hodnoty, ktorá sa potom uloží do databázy.
- `rows`: Udáva koľko riadkov sa uloží do databázy.

```
def create_rrd_db():
startTime = int(time.time()) - 86400
print(f'{startTime}')
rrdtool.create(rrd_db_name) ,
'--start', str(startTime),
'--step', '10',
'DS:luminosity:GAUGE:25:U:U' ,
'DS:noise:GAUGE:25:U:U' ,
"RRA:AVERAGE:0.5:6:40320",
```

Výpis 3: Vytvorenie vlastnej databázy v RRDtool

`Starttime` hodnota definuje časové intervaly, nastavenie hodnoty na 86400 definuje čas čas v sekundách medzi data pointami. V tomto prípade hodnota predstavuje rovnicu 24h*60minút*60sekúnd so začiatkom `now`.
`-step` základný inerval, v ktorom sa budú dáta prenášať do RRD. RRA definuje Round robin archív, CF je typu `AVERAGE`, `xxf` konsolidačný interval je nastavený na hodnoty 0.5 v 2 krokoch a 43800 stĺpcoch.

Funkcia update

Slúži k naplneniu databáze, ktorú sme si vytvorili. Dáta sú v databáze automaticky ukladané podľa zadaných parametrov pri vytvorení databáze.

- `filename` je názov RRD súboru, ktorý chceme aktualizovať.
- `[-template|-t ds-name[:ds-name]...]` v pôvodnom nastavení funkcia UPDATE očkáva, že vstupné dáta budú v rovnakom poradí ako sú DS definované v RRD. Pomocou voľby DS si môžeme určiť, ktoré DS budú aktualizované a v akom poradí. Pokiaľ uvedieme v templates DS, ktoré v RRD súbore nie sú, tak sa proces aktualizácie zastaví a vypíše chybu.
- `[-daemon address] [-]` pri použití RRDtool deamona sa k nemu pokúsi pripojiť. Pokiaľ je spojenie úspešné, tak budú hodnoty zaslané na adresu daemona k ďalšiemu spracovaniu.
- `N|timestamp:value[:value...]` - je hodnota, ktorá určuje počet sekúd ubehnutých od 1.1.1970. Za dvojbodkami zapisujeme hodnoty v rovnakom poradí ako ich máme definované v RRD súbore.

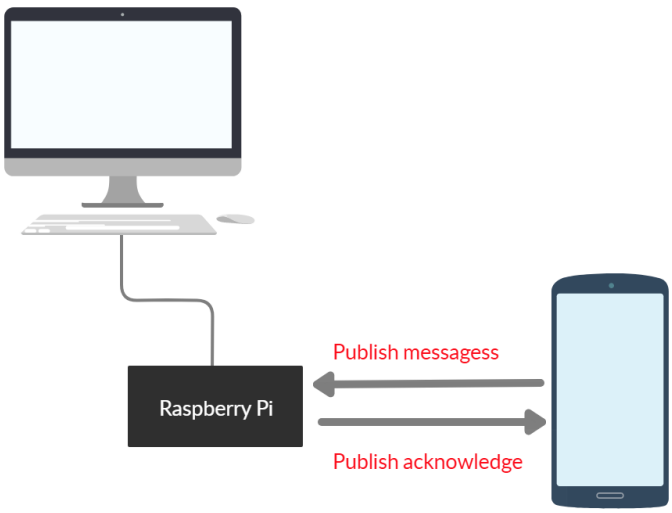
```
rrdtool.update(rrd_db_name,'%d:%f:%f:%f' % (now, lastLuminosity, lastNoise))
print('%d:%f:%f:%f' % (now, lastLuminosity, lastNoise)) #Note: sent to the
    database
```

Výpis 4: Ukážka vytvorenia databázy RRDtool

4 Návrh riešenia automatizovaného zberu a prenosu dát

4.1 Návrh riešenia

Návrh riešenia pozostáva z myšlienky využitia mikropočítača RaspberryPi 4, ktorý predstavuje centralizovaný bod - *broker*.
Rasspery Pi 4 disponuje štvorjadrovým procesorom Cortex-A72 s maximálnym taktom 1,5 GHz. Operačná pamäť má kapacitu 2GHz. Základná doska pozostáva zo štyroch USB portov(2x 2.0, 2x 3.0), Ethernetového portu, dvoch micro HDMI rozhraní a napájania USB-C. Rovnako tiež obsahuje microSD kartu o veľkosti 32GB s predinštalovaným operačným systémom Raspbian. K mikropočítaču je pripojený monitor Samsung S22F350 pomocou microHDMI kábla a súčasne tiež klávesnica s myšou.
Klienta - *publisher* - predstavuje mobilné zariadenie Xiaomi Mi A3 s operačnou pamäťou 4GB a vnútorným úložiskom o veľkosti 64GB.

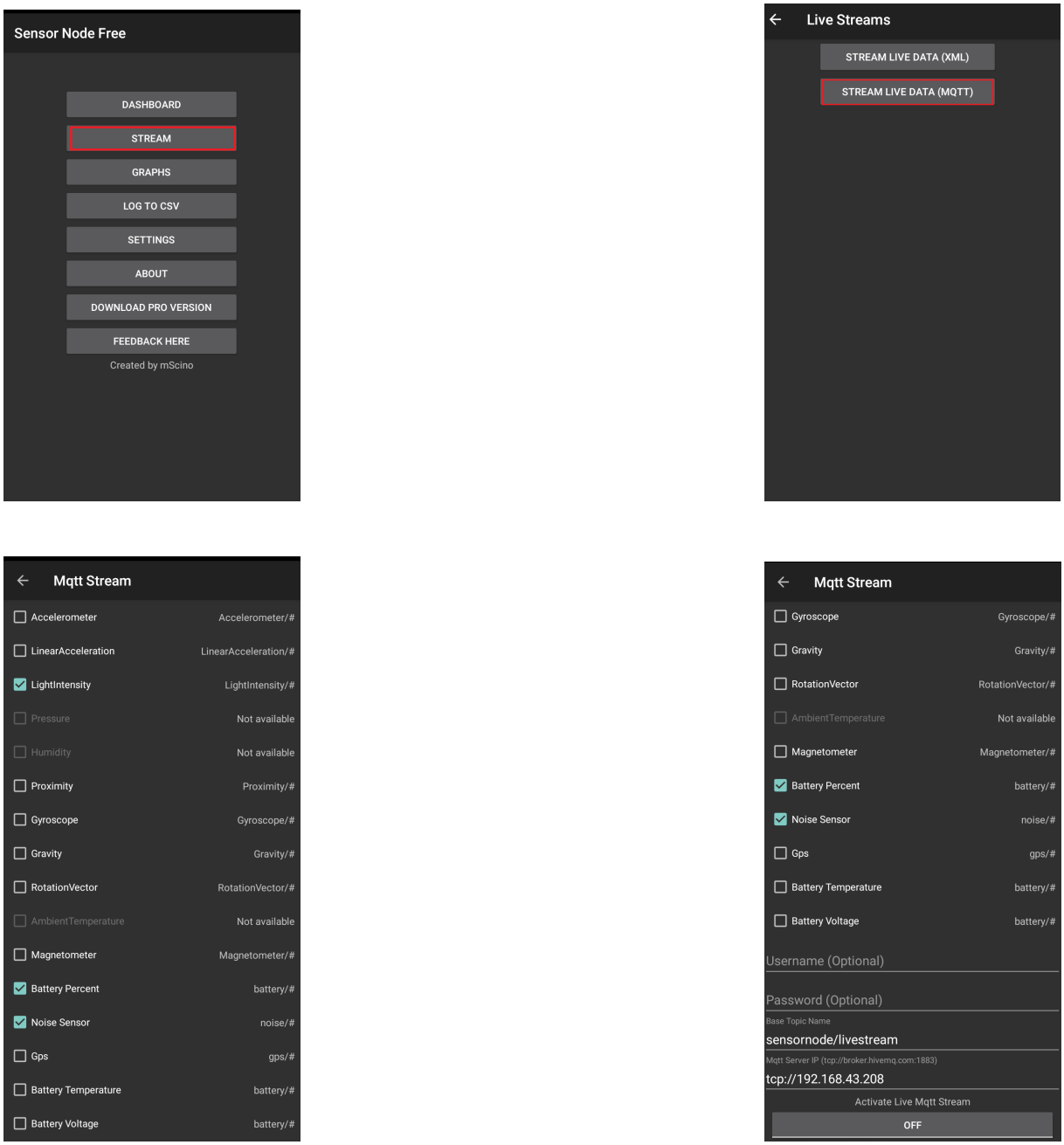


Obrázek 8: Model zapojenia.

V tomto návrhu posiela mobilný klient - publisher - MQTT správy prostredníctvom aplikácie z tretej strany, tie sú následne spracúvavané Raspberry Pi, ktorý funguje ako broker.

4.2 Technické a programové vybavenie na realizáciu návrhu

Vo svojej práci som využila verejne dostupnú aplikáciu **Sensor Node Free** [13] z obchodu Google play. Aplikácia bola navrhnutá tak, že umožňuje odosielať dáta zo senzorov mobilného zariadenia na adresu brokera a súčasne nastaviť meno a heslo pre užívateľa. Súčasne poskytuje možnosť vykresľovania grafov v reálom čase a ukladanie dát do CSV.



Obrázek 9: Nastavenie aplikácie Sensor Node.

4.3 Konfigurácia MQTT brokera

Existuje viacero distribúcií MQTT brokera. Do svojej práce som využila open source MQTT brokera - Eclipse Mosquitto(EPL/EDL licensed), ktorý implemetuje MQTT verzie 3.1.0, 3.1.1 a version 5.0. Je napísaný v jazyku C (autor Roger Light) a dostupný pre operačné systémy Windows a Linux.

```
#inštalácia Mosquitto Brokera na Raspberry Pi
pi@raspberry:~ $ sudo apt update
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients
```

Výpis 5: Inštalácia Mosquitto Brokera na Raspberry Pi.

Inštaláciu potvrdíme **Y** a **ENTER** klávesou. A na automatické naštartovanie Mosquitta spustíme service `sudo systemctl enable mosquitto.service`.

```
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-05-11 01:17:05 CEST; 3 days ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
 Main PID: 475 (mosquitto)
    Tasks: 1 (limit: 4035)
   Memory: 912.0K
   CGroup: /system.slice/mosquitto.service
           └─475 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

Obrázek 10: Mosquitto service.

Pomocou príkazu `mosquitto -v` potvrdíme, že inštalácia prebehla v poriadku a cez `sudo service mosquitto stat` si potvrdíme, že service je aktívny.

4.4 MQTT Subscriber

V návrhu riešenia absentuje prítomnosť ďalšieho klienta, ktorý v MQTT funguje ako *subscriber*. K tomu som využila MQTT knižnicu v jazyku Python, ktorá poskytuje klientskú triedu, ktorá umožňuje aplikáciám sa pripojiť na *brokera* pre publikovanie správ, ale i odber správ či príjem publikovaných správ.
Knižnica MQTT Paho Client [12] knižnice mi umožnila zobraziť správy, ktoré boli odoslané z mobilného klienta na brokera.
Adresa brokera predstavuje IP adresu Raspberry Pi a port 1883 predstavuje port, na ktorom MQTT pracuje. Pri úspešnom vytvorení spojenia sa v terminále objaví *"Connected to broker"* a pri neúspšenom nadviazaní spojenia *"Connection failed"*. MQTT správy, ktoré odosiela mobilný klient predstavujú správy o stave batérie, jase a hluku v okolí.

```
#Ukáža vytvorenia python mqtt paho clienta
from paho.mqtt.client import Client
broker_address= "192.168.43.208"
port = 1883

def on_connect(client, userdata, flags, rc):
if rc == 0:
print("Connected to broker")
else:
print("Connection failed")

def on_message(client, userdata, message):
try:
global lastLuminosity
global lastNoise

print(f'log: {datetime.datetime.now()} - Message received {message.topic}: {str
    (message.payload.decode("utf-8"))}')
now = int(time.time())
payload = float(message.payload)

if message.topic == "sensornode/livestream/LightIntensity/x":
print(f'\033[92mLuminosity {payload} lux\033[0m')
lastLuminosity = payload

if message.topic == "sensornode/livestream/noise/decibels":
print(f'\033[92mNoise {payload} dBA\033[0m')
lastNoise = payload

time.sleep(5)

except Exception as e:
print(e)

client = Client("Python")
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log
```

```
client.connect(broker_address,port=port)
client.subscribe('sensornode/livestream/#')
client.loop_forever()
```

Výpis 6: MQTT Paho Client

5 Vykresľovanie v programe CACTI

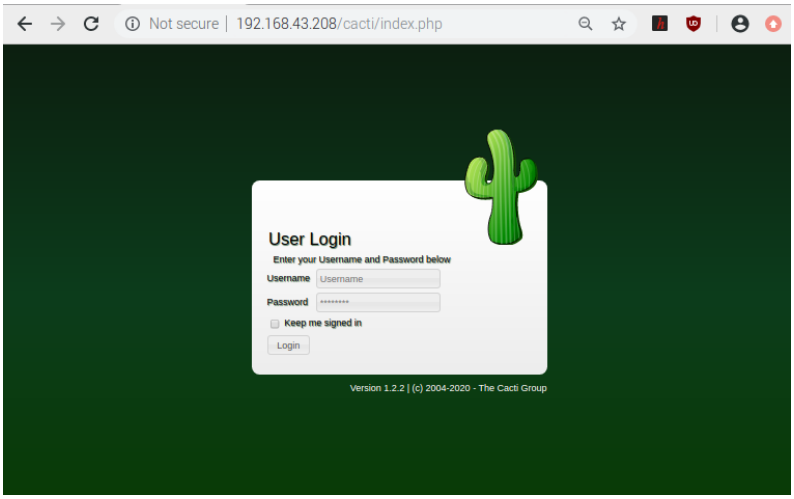
Cacti je monitorovací tool založený na **RRD tool**, ktorý umožňuje ukladanie dát a vykresľovanie grafov v reálnom čase. Cacti obsahuje prieskumník, pokročilé šablóny, možnosť administrácie užívateľských účtov a práv. Bežná prevádzka sieťových zariadení je vykresľovaná za pomoci SNMP sieťového protokolu.
Projekt Cacti vyšiel v roku 2001, ktorý spustil Ian Berry. Pôvodná myšlienka bola využiť RRDtool, tak aby práca s nim bola jednoduchšia.

5.1 Konfigurácia CACTI

```
#Inštalácia Cacti
sudo apt-get update
sudo apt-get upgrade

sudo apt-get install apache2 php5 mysql-client mysql-server php5-mysql php5-
    snmp php5-gd php5-ldap rrdtool snmp snmpd -y
sudo apt-get install cacti -y
```

Výpis 7: Inštalácia Cacti

Po úspešnom nainštalovaní MY-SQL servera je užívateľ vyzvaný k zadaniu hesla pre MYQSL admina.
Pri inštalácií Cacti budeme vyzvaní k výberu webservera `apache2` a rovanko zadaniu hesla pre MYSQL usera. Posledným krokom je zadanie hesla pre administrátora databáze. Po úspešnej inštalácií sa užívateľ môže prehliadačom prihlásiť do Cacti cez `http://HOSTNAME/cacti`. Hostname v našom prípade predstavuje IPadresu RaspberryPi.citeTeach



Obrázek 11: Prihlasovacia obrazovka Cacti.

5.2 Povolenie a nastavenie SNMP

Aby bolo možné povoliť SNMP na Raspberry (alebo na akomkoľvek inom Linuxovom systéme) je nutné, aby boli najskôr nakonfigurované služby SNMP.
SNMP je protokol aplikačnej vrstvy, ktorý slúži na monitorovanie a správu zariadení v IP sieti.

Architektúra rozpoznáva z dvoch komponentov:

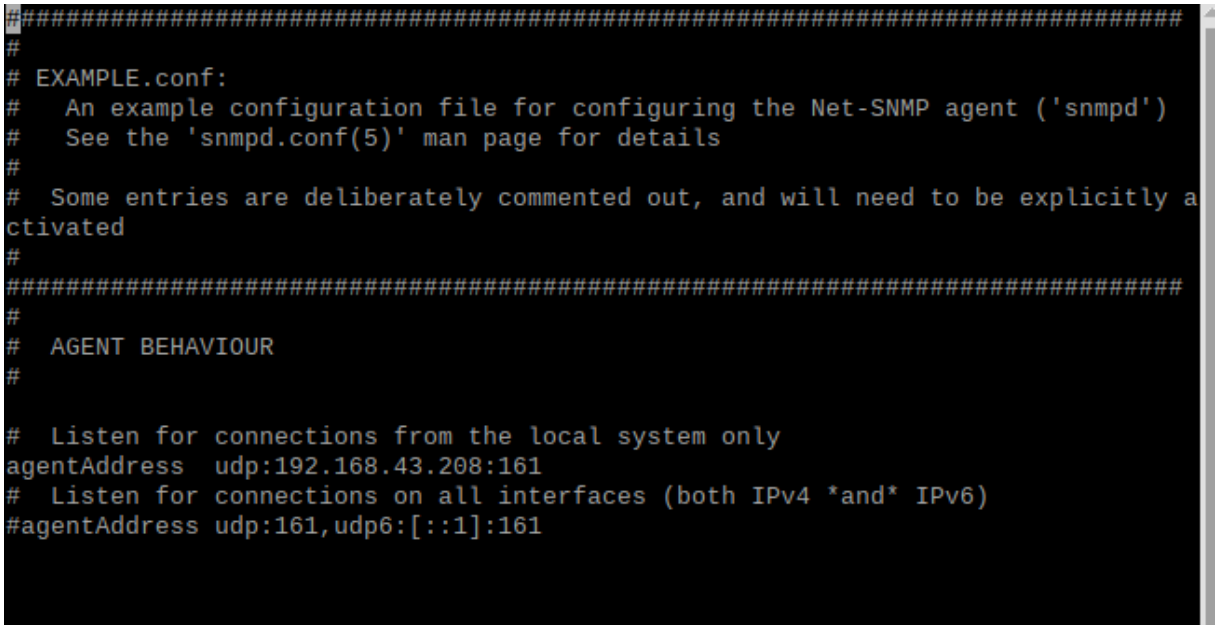
- SNMP Agent je software bežiaci vo vnútri zariadení, ktoré sú pripojené k sieti. SNMP Agent pracuje na porte UDP 162.
- SNMP Manager (NMS) je systém, ktorý zodpovedá za komunikáciu s pripojenými SNMP agentami. SNMP manager pracuje na porte UDP 161.
- MIB je databáza je hierarchickej štruktúry, ktorá sa používa k správe sietí. Každá položka databázy je definovaná pomocou identifikátora objektu(OID).

Pri typickom použití protokolu SNMP má jeden alebo viac administratívnych počítačov - managerov - úlohu monitorovať alebo spravovať skupinu hostov. Každý riadený systém vykonáva softvérovú súčasť nazývanú agent, ktorý hlási informácie cez SNMP správcovi.
Management information base(MIB) popisuje štruktúru spravovaných dát, kde sa využíva hierarchický systém obsahujúci identifikátori objektov(OID).

```
#Nastavenie SNMP
sudo apt install -y snmpd snmp
sudo systemctl stop snmpd
```

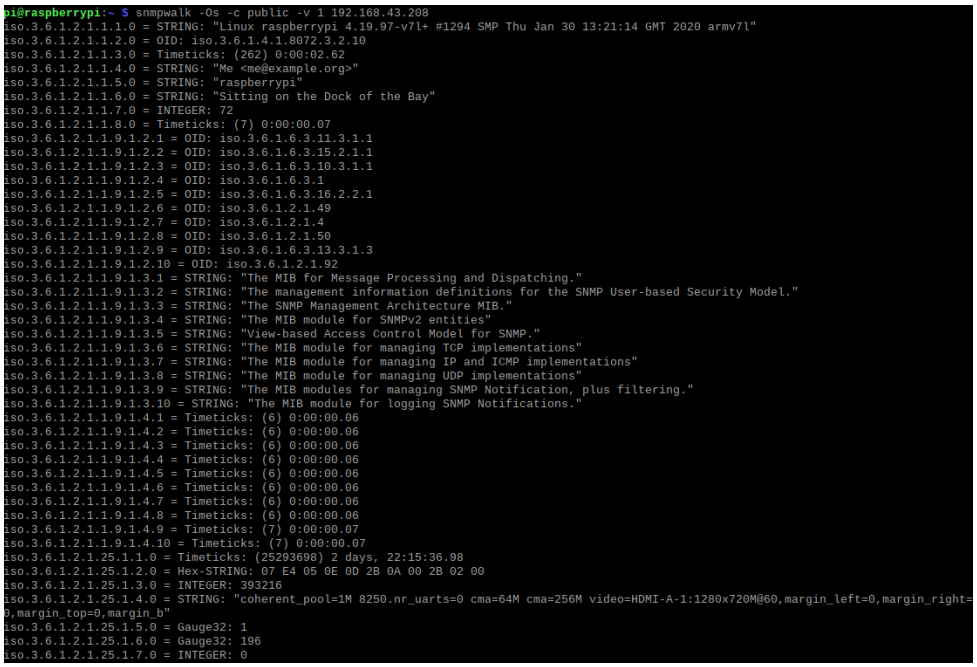
Výpis 8: Nastavenie SNMP

V zložke /etc/snmp/snmpd.conf modifikujeme adresu hosta na adresu Raspberry.



Obrázek 12: Nastavenie SNMP agenta.

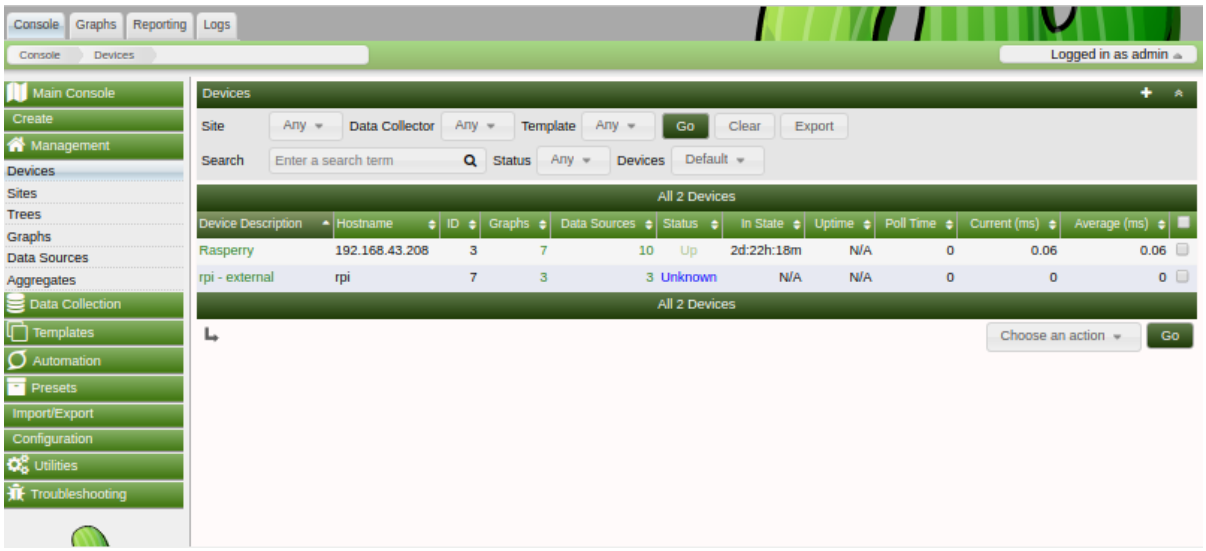
Po nastavení potrebnej konfigurácie, reštartujeme service cez `service snmpd restart`. Overenie funkčnosti môžeme vykonať cez `snmpwalk -Os -c public -v 1 [IP]`.



Obrázek 13: Prihlasovacia obrazovka Cacti.

5.3 Nastavenie zariadení v CACTI

V zložke `Create/New Devices`, môže užívateľ pridávať zariadenia, ktoré chce monitorovať. `Managment/Devices` mu umožňuje upravovať nastavenia pre jednotlivé zariadenia, ktoré si pridal.



Obrázek 14: Cacti Managment zariadení.

V tomto prípade sú pridané dve zariadenia, Rasberry a rpi-external. Zariadenie Rasperry je monitorované cez SNMP a zariadenie rpi-external je monitorované cez databázu RRDtool.

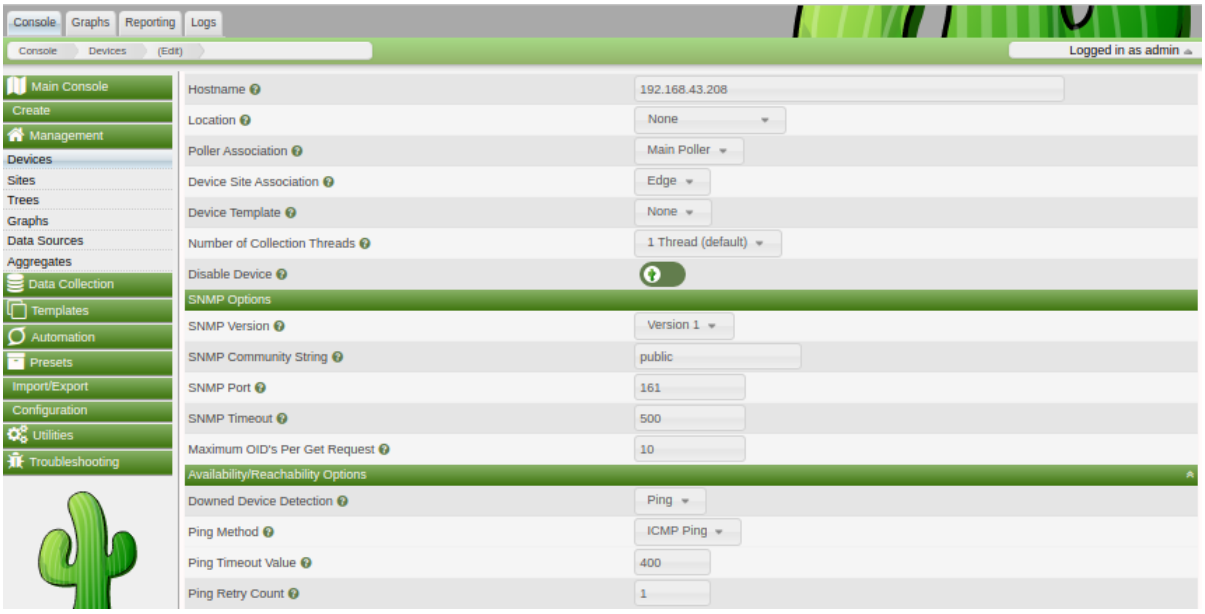
Nové zariadenie definujeme cez parametre `Description`, ktoré určuje jeho popis, `Hostname` môže byť string alebo IP adresa, ktorá odkazuje na dané zariadenie. V prípade, že chceme monitorovať zariadenie cez SNMP definujeme jeho parametre v `SNMP Options` - definujeme verziu, port a string. V prípade, že chceme monitorovať dostupnosť zariadenia vyplníme `Availability Options`.

V druhom prípade máme zariadenie, pre ktoré chceme vykresľovať grafy externe plnenou databázou RRD.
V `Description` definujeme, že zariadenie je externého typu: rpi - external. V tomto prípade je dôležité, aby `Device Template` bol definovaný ako NONE.

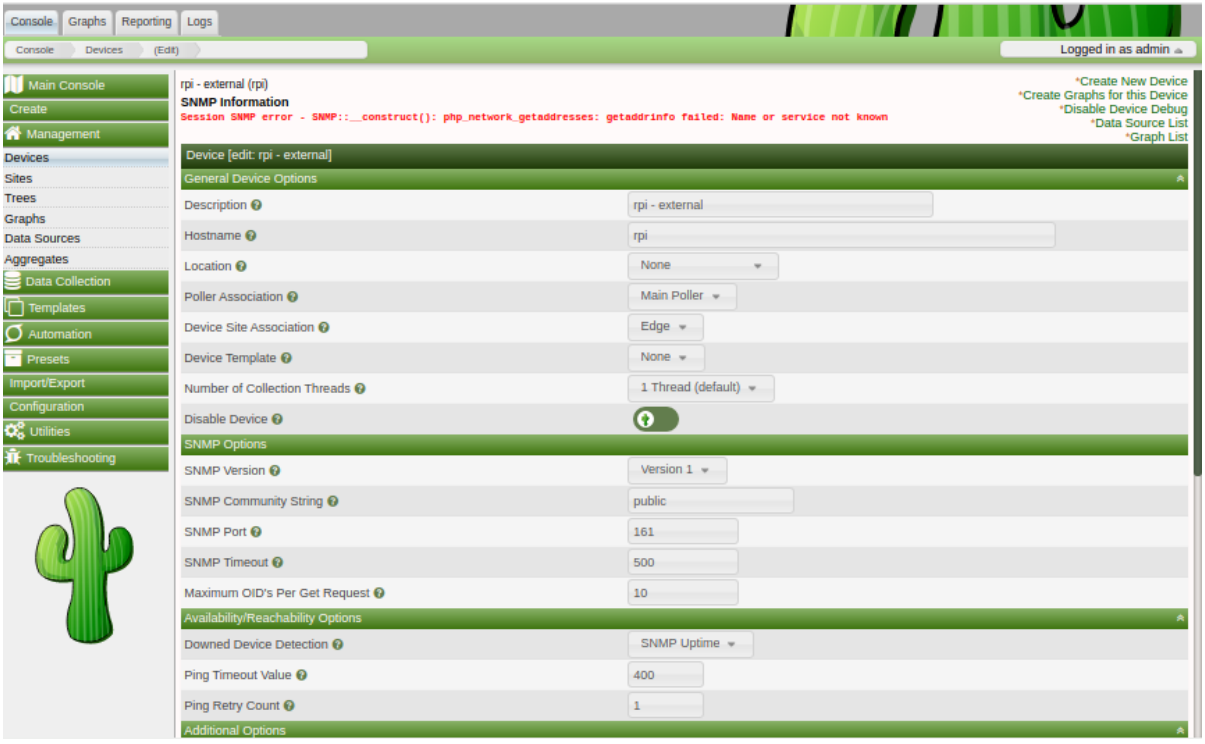
Cacti má preddefinované Templates, ktoré slúžia ako šablóny pre monitoring zariadení. Cacti však ponúka možnosť definovať si vlastné Grafy v zložke `Templates/Graphs`.

V zložke Templates/Graph som definovala Items, ktoré chcem vykresliť do Externého grafu - Luminosity, Noise rovnako som definovala hodnotu, ktorú chcem vykresľovať a farbu, ktorou sú jednotlivé itemy reprezentované.

Rovnako dôležité je tiež definovať v Zložke *Managment/Data Source* celú cestu k databáze a jednotlivé itemy, ktoré sú obsiahnuté v databáze.



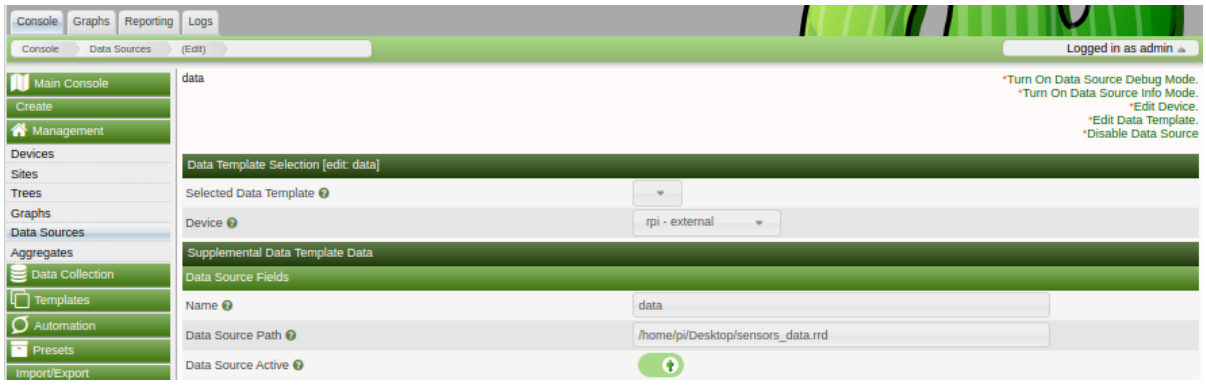
Obrázek 15: Nastavenie zariadenia Rasperry.



Obrázek 16: Nastavenie zariadenia Rpi.

Že nám monitoring cez snmp funguje môžeme zistiť pri vykreslení grafov pre zariadenie Raspberry. V zložke `Graphs` vidíme všetky dostupné grafy pre všetky zariadenia, ktoré sme definovali v Cacti.

Obrázek 17: Nastavenie parametrov grafu pre rpi.



Obrázek 18: Nastavenie Data Source.

Z dostupných grafov som si vybrala dva - využitie pamäte, oneskorenie pingu a primerné vyťaženie systému. Z dostupných grafov vidíme, že monitoring cez snmp funguje.

6 Overenie funkčnosti

V práci som pracovala s dvoma scriptami, create DB skriptom, ktorý vytvoril databázu a Run script, ktorý ju plnil databázu dátami z mobilného klienta.

```
#Create db script
from paho.mqtt.client import Client
import rrdtool, sys, time, datetime

rrd_db_name = "sensors_data.rrd"

def create_rrd_db():
startTime = int(time.time()) - 86400
print(f'{startTime}')
rrdtool.create(rrd_db_name ,
'--start', str(startTime),
'--step', '10',
'DS:luminosity:GAUGE:25:U:U' ,
'DS:noise:GAUGE:25:U:U' ,
'DS:battery_level:GAUGE:25:U:U',
"RRA:AVERAGE:0.5:6:40320",
"RRA:MAX:0.5:1:40320",
"RRA:MIN:0.5:1:40320",
)
create_rrd_db()
```

Výpis 9: Create DB script

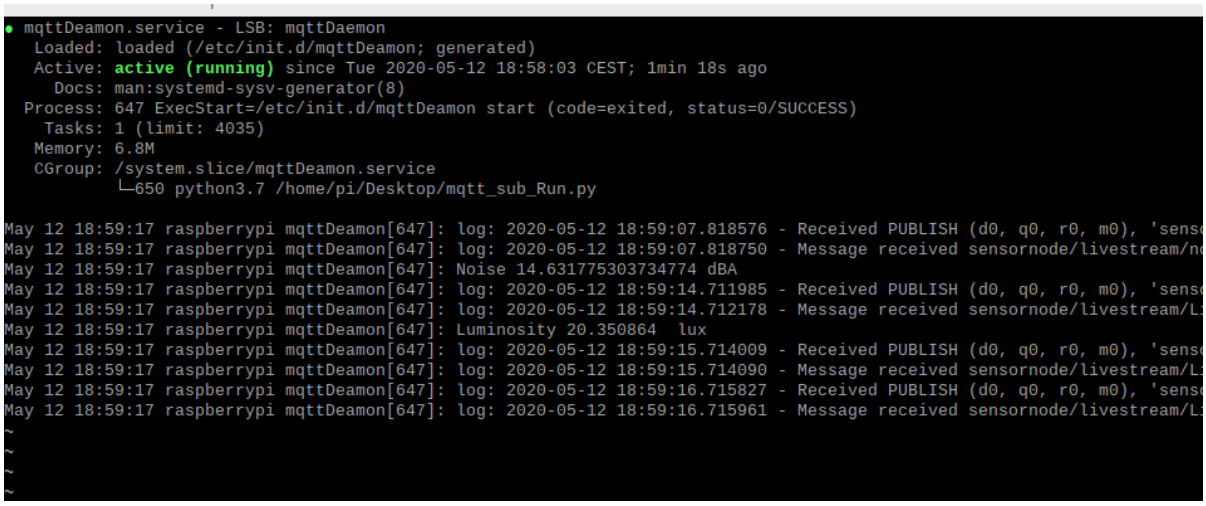
Create db skript založí novú databázu, k tomuto skriptu pristupujeme samostatne a to len v prípade, keď chceme zmeniť parametre databázy.
Rozdelením skriptu na dva menšie zamedzíme tomu, aby sa nám prepisovala databáza a tak aby sme neprišli o doposiaľ zozbierané dáta.
Run skript beží ako service a v prípade, že chceme meniť parametre v Create skripte, musíme tento service zastaviť a opätovne reštartovať.

```
#Run script
from paho.mqtt.client import Client
import rrdtool, sys, time, datetime

broker_address= "192.168.43.208"
port = 1883

rrd_db_name = "/home/pi/Desktop/sensors_data.rrd"

lastLuminosity = 0
lastNoise = 0
lastBattery = 0

def on_connect(client, userdata, flags, rc):
if rc == 0:
print("Connected to broker")
else:
print("Connection failed")

def on_log(client, userdata, level, buf):
print(f'log: {datetime.datetime.now()} - {buf}')

def on_message(client, userdata, message):
try:
global lastLuminosity
global lastNoise
global lastBattery

print(f'log: {datetime.datetime.now()} - Message received {message.topic}: {str
    (message.payload.decode("utf-8"))}')
now = int(time.time())
payload = float(message.payload)
if message.topic == "sensornode/livestream/LightIntensity/x":
print(f'\033[92mLuminosity {payload} lux\033[0m')
lastLuminosity = payload

if message.topic == "sensornode/livestream/noise/decibels":
print(f'\033[92mNoise {payload} dBA\033[0m')
```

```
lastNoise = payload
rrdtool.update(rrd_db_name,'%d:%f:%f:%f' % (now, lastLuminosity, lastNoise,
    lastBattery))
time.sleep(1)

except Exception as e:
print(e)

client = Client("Python")
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log

client.connect(broker_address,port=port)
client.subscribe('sensornode/livestream/#')
client.loop_forever()
```

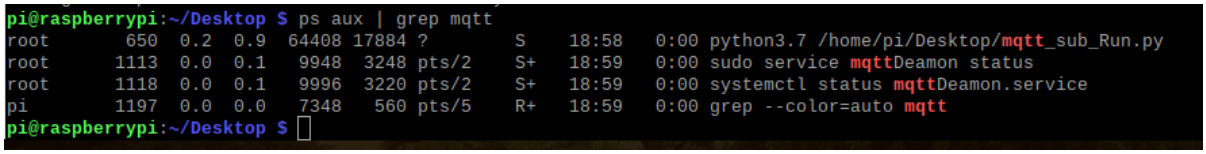
Výpis 10: Run script

To, či service funguje overíme pomocou **sudo service [nazov servisu] status**. V prípade, že chceme modifikovať Create skript, musíme zastaviť tento service cez **sudo service [nazov servisu] stop** a následne ho obnoviť **sudo service [nazov servisu] reload**.



Obrázek 19: Service status.

Cez príkaz **ps aux | grep mqtt** uvidíme tento service ako bežiaci.



Obrázek 20: Service status.

Cez príkaz **watch n -1 rrdtool info /home/pi/Desktop/runskript.py** ako sa databáza updatuje a ako nám hodnoty rastú, prípadne klesajú.

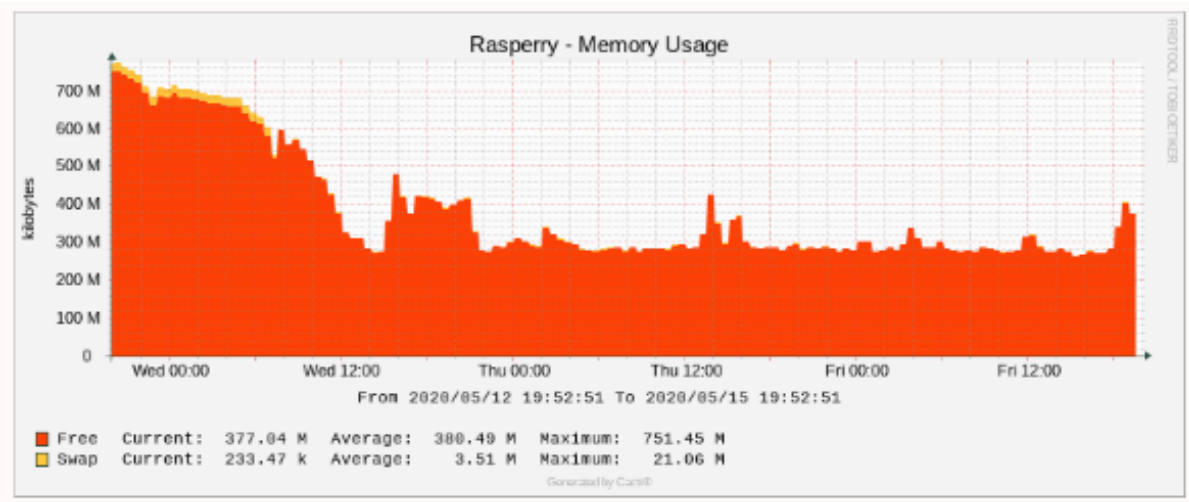
```
Every 1.0s: rrdtool info /home/pi/Desktop/sensors_data.rrd

filename = "/home/pi/Desktop/sensors_data.rrd"
rrd_version = "0003"
step = 10
last_update = 1589346089
header_size = 1180
ds[luminosity].index = 0
ds[luminosity].type = "GAUGE"
ds[luminosity].minimal_heartbeat = 25
ds[luminosity].min = NaN
ds[luminosity].max = NaN
ds[luminosity].last_ds = "146.666580"
ds[luminosity].value = 1.3199992200e+03
ds[luminosity].unknown_sec = 0
ds[noise].index = 1
ds[noise].type = "GAUGE"
ds[noise].minimal_heartbeat = 25
ds[noise].min = NaN
ds[noise].max = NaN
ds[noise].last_ds = "-6.584484"
ds[noise].value = -5.9260356000e+01
ds[noise].unknown_sec = 0
ds[battery_level].index = 2
ds[battery_level].type = "GAUGE"
ds[battery_level].minimal_heartbeat = 25
ds[battery_level].min = NaN
ds[battery_level].max = NaN
ds[battery_level].last_ds = "0.000000"
ds[battery_level].value = 0.0000000000e+00
ds[battery_level].unknown_sec = 0
rra[0].cf = "AVERAGE"
```

Obrázek 21: RRD tool info.

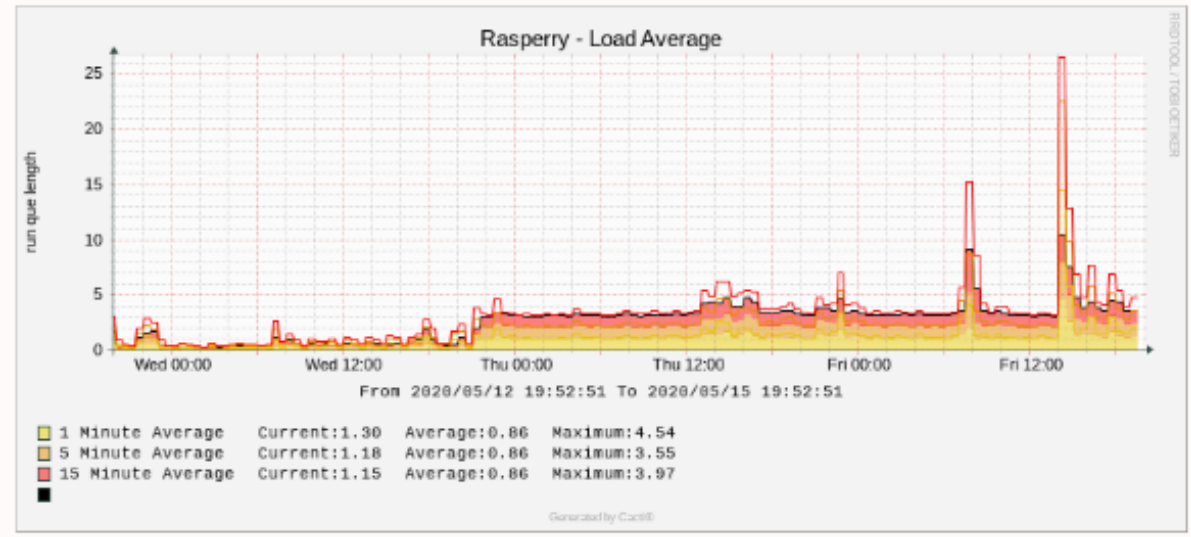
6.1 Vykreslenie jednotlivých grafov

Časový interval troch dní

V nasledujúcich obrázkoch (Obrázok č.22 a Obrázok č.23) môžeme vidieť využitie pamäte a priemerné zaťaženie systému RasperryPi v časovom intervale troch dní.

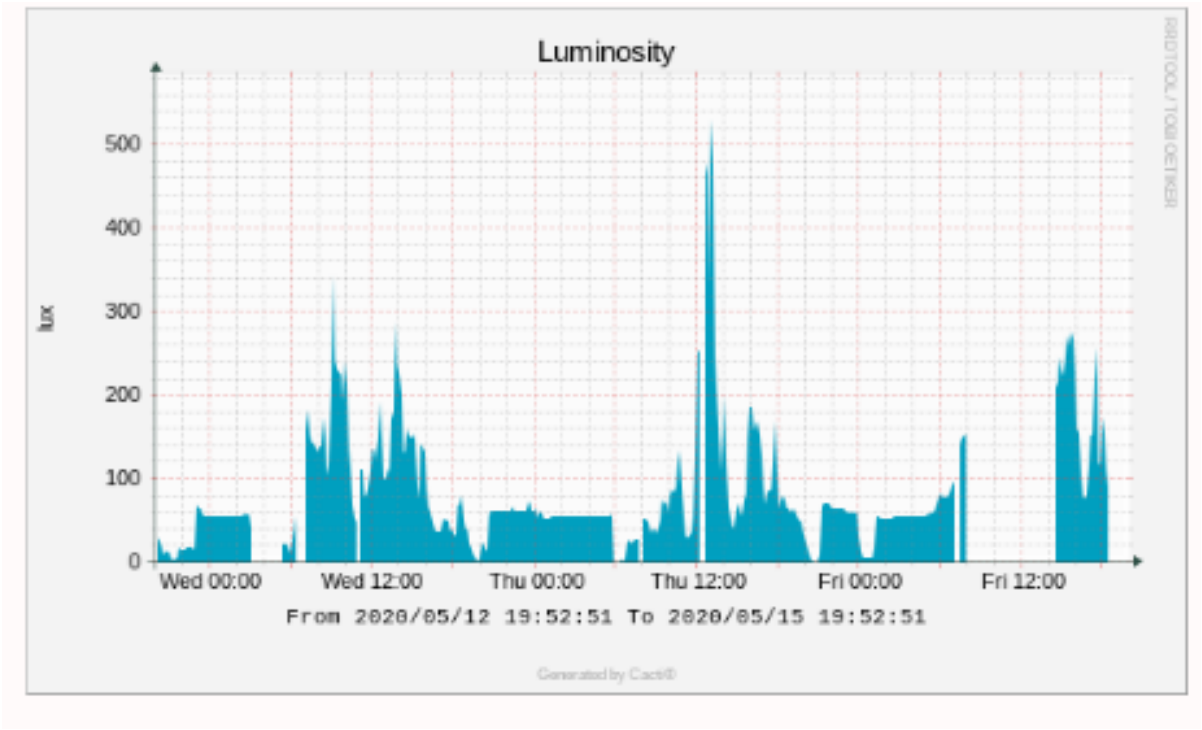


Obrázek 22: Využitie pamäte.

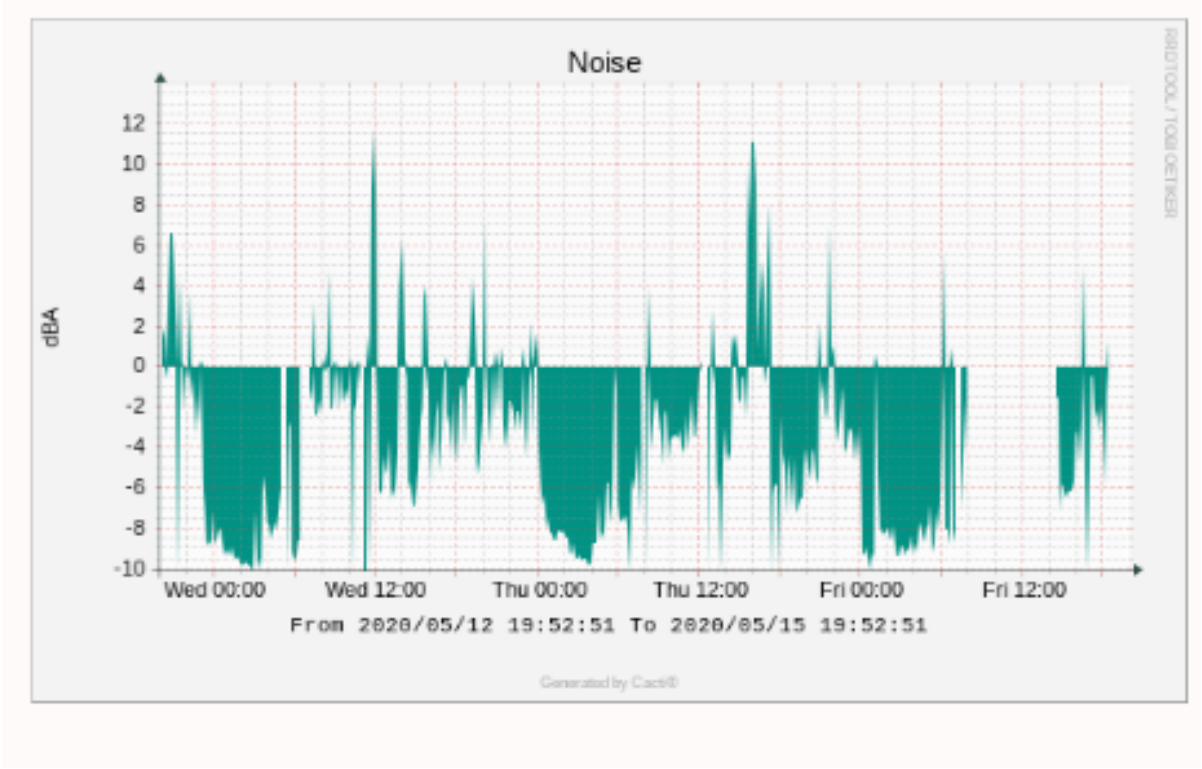


Obrázek 23: Vyťaženie pamäte.

Na obrázkoch (Obrázok č.24 a Obrázok č.25) môžeme vidieť úspešne vykreslenie dát zo senzorov mobilného telefónu. Obrázek č. 24 vyobrazuje namerané hodnoty svietivosti v intervale troch dní, a Obrázek č.25 zobrazuje namerané hodnoty úrovne hluku. V oboch prípadoch nastala nedostupnosť dát v rovnakom časovom úseku, kedy sa ukladanie dát nevykonávalo. Tento jav bol spôsobený nedostupnosťou zariadenia v čase.



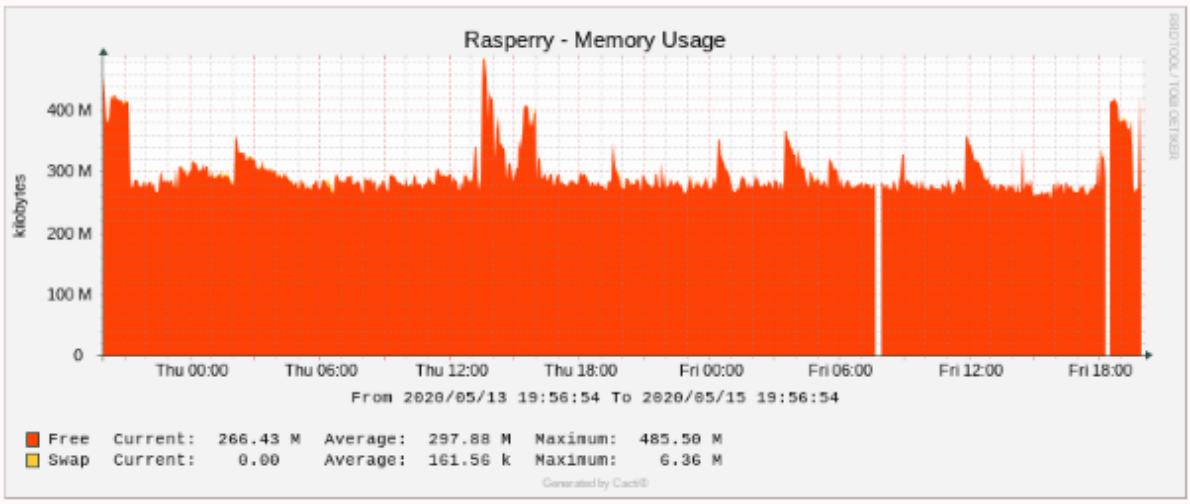
Obrázek 24: Svietivosť.



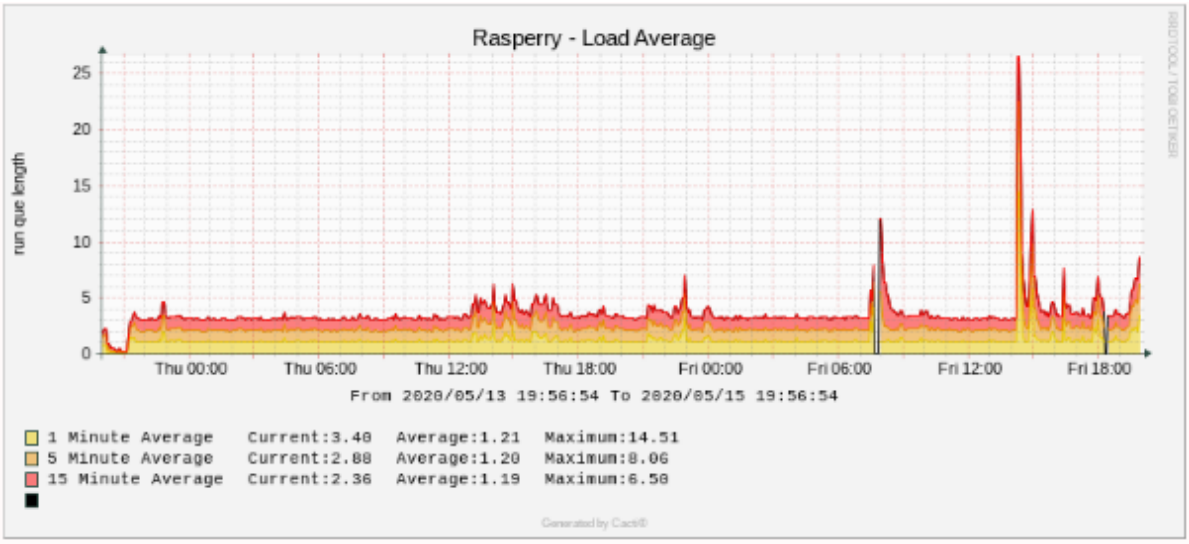
Obrázek 25: Úroveň hluku.

Časový interval dvoch dní

V nasledujúcich obrázkoch (Obrázok č. 26 a Obrázok č.27) môžeme vidieť využitie pamäte a priemerné zaťaženie systému RasperryPi v časovom intervale dvoch dní. V oboch prípadoch môžeme vidieť nesúvislý graf a výpadky.

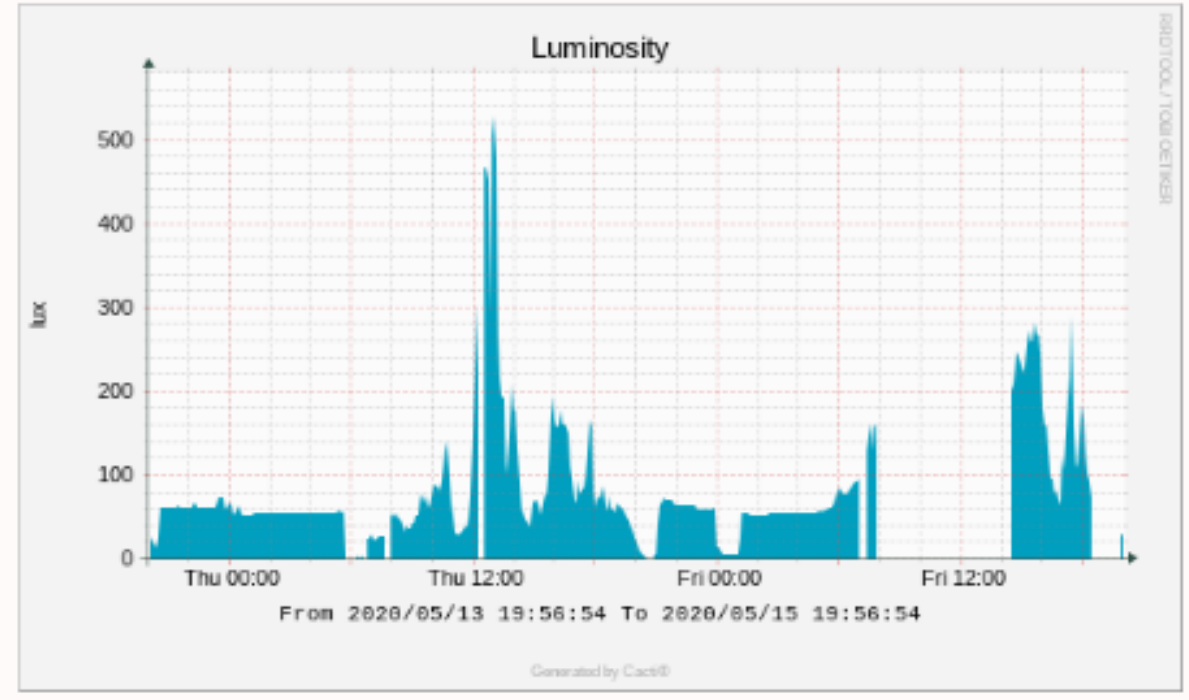


Obrázek 26: Využitie pamäte.

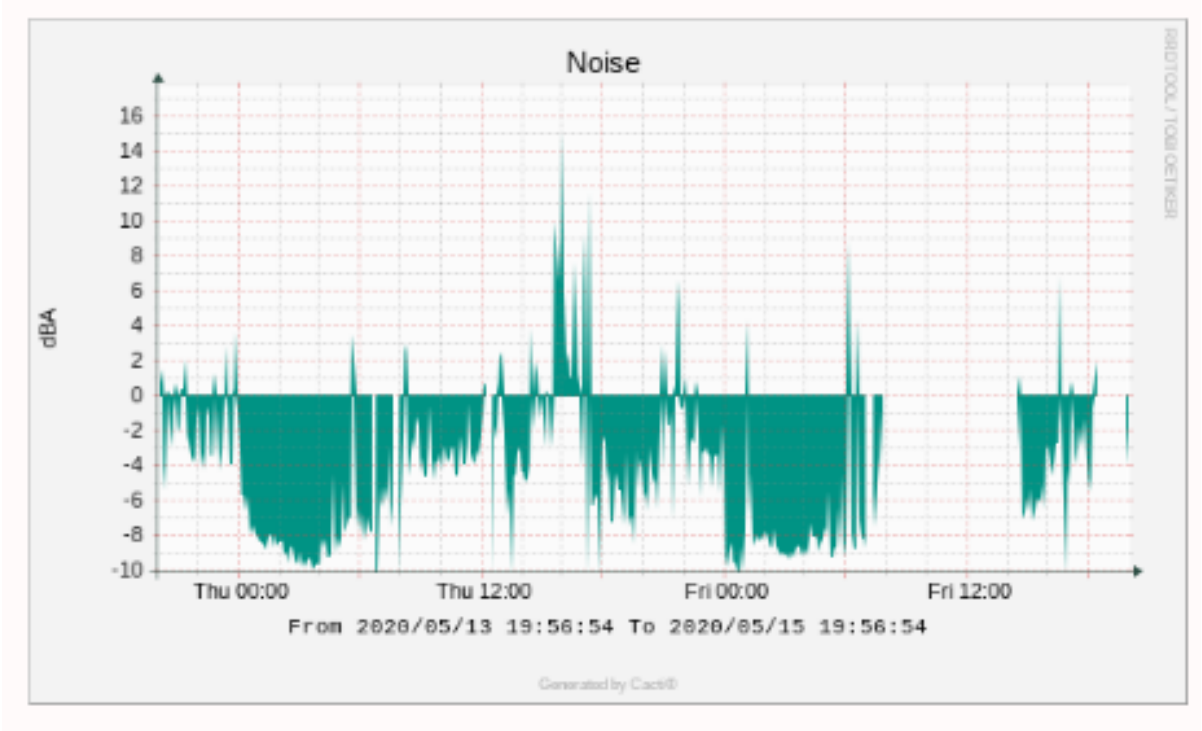


Obrázek 27: Vyťaženie pamäte.

Na obrázkoch (Obrázok č.28 a Obrázok č.29) zobrazujeme namerané hodnoty svietivosti a úrovne hluku v intervale dvoch dní.



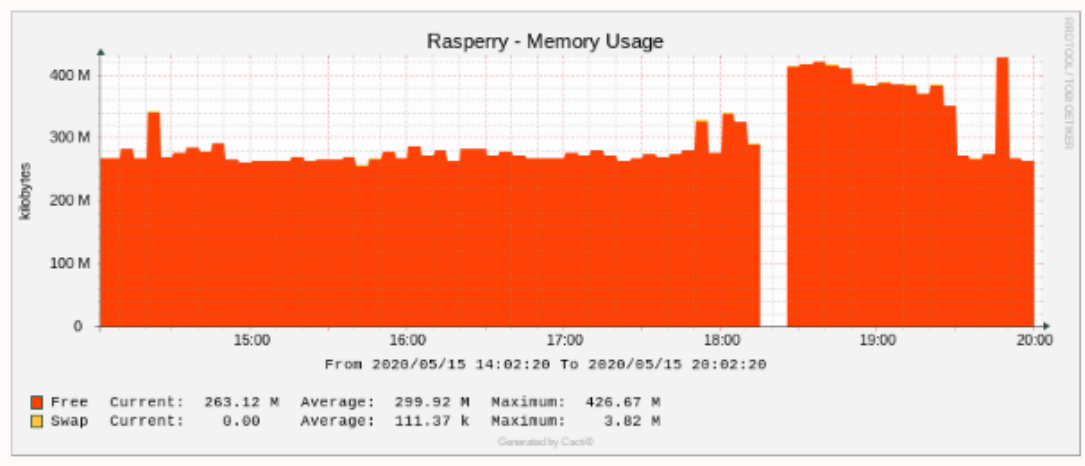
Obrázek 28: Luminosity.



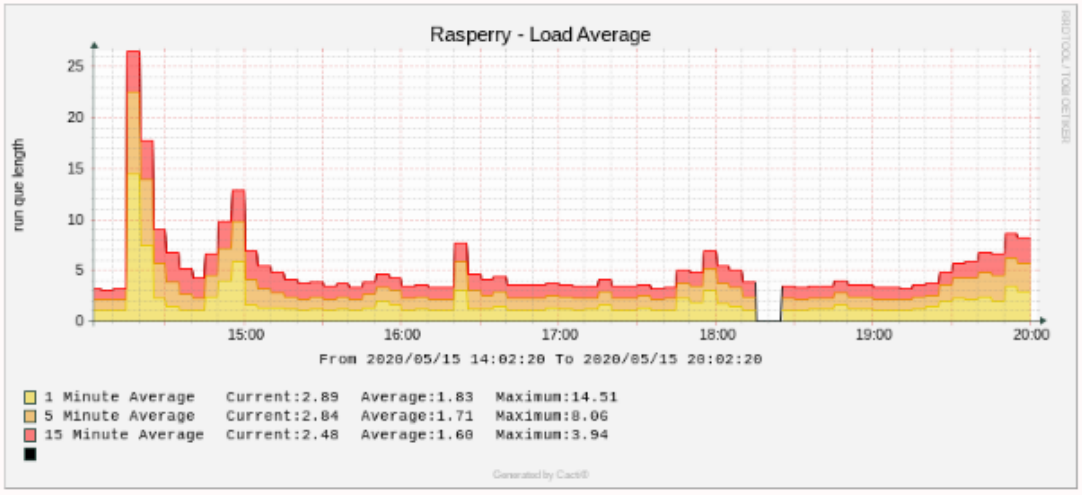
Obrázek 29: Úroveň hluku.

Interval šiestich hodín

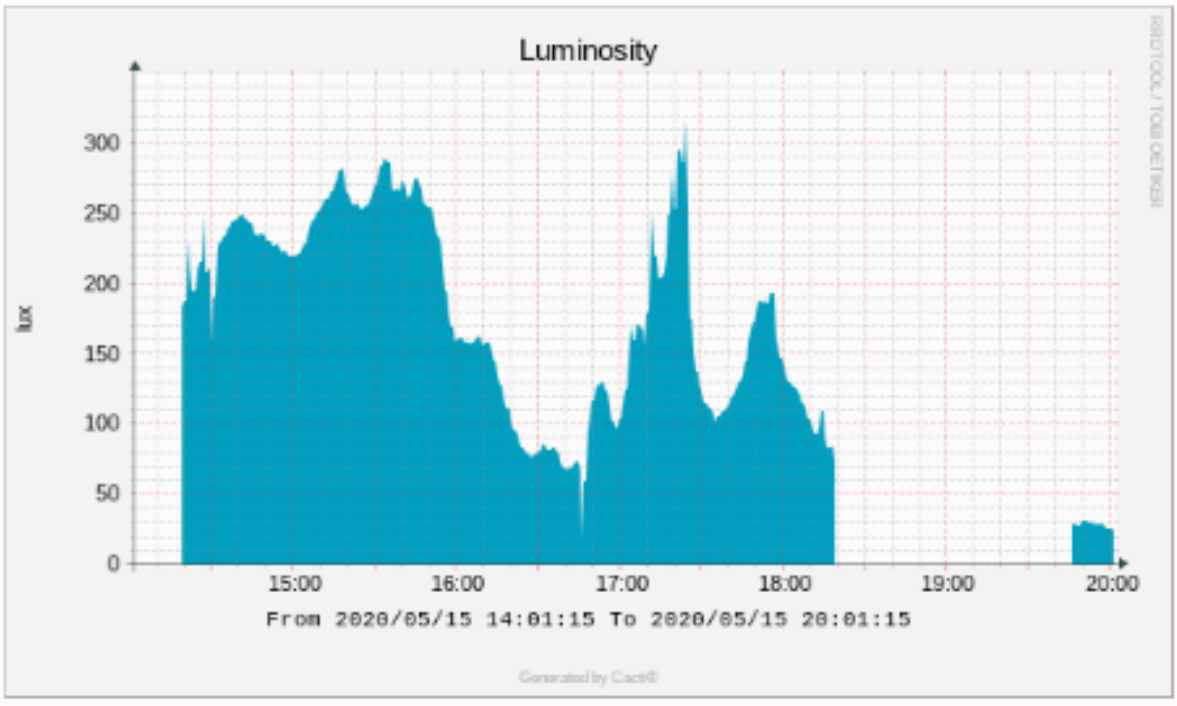
V intervale šiestich hodín môžeme pozorovať, že nastalov vyťaženie pamäte a výpadok. Rovnako to bolo aj v prípade merania svietivosti a úrovne hluku. V tomto prípade však môžeme pozorovať, že nedostupnosť dát pretrvávala v dlhšom úseku, čo bolo spôsobené ľudským faktorom. V tomto prípade sa užívateľ vzdialil s mobilným zariadením od brokera.



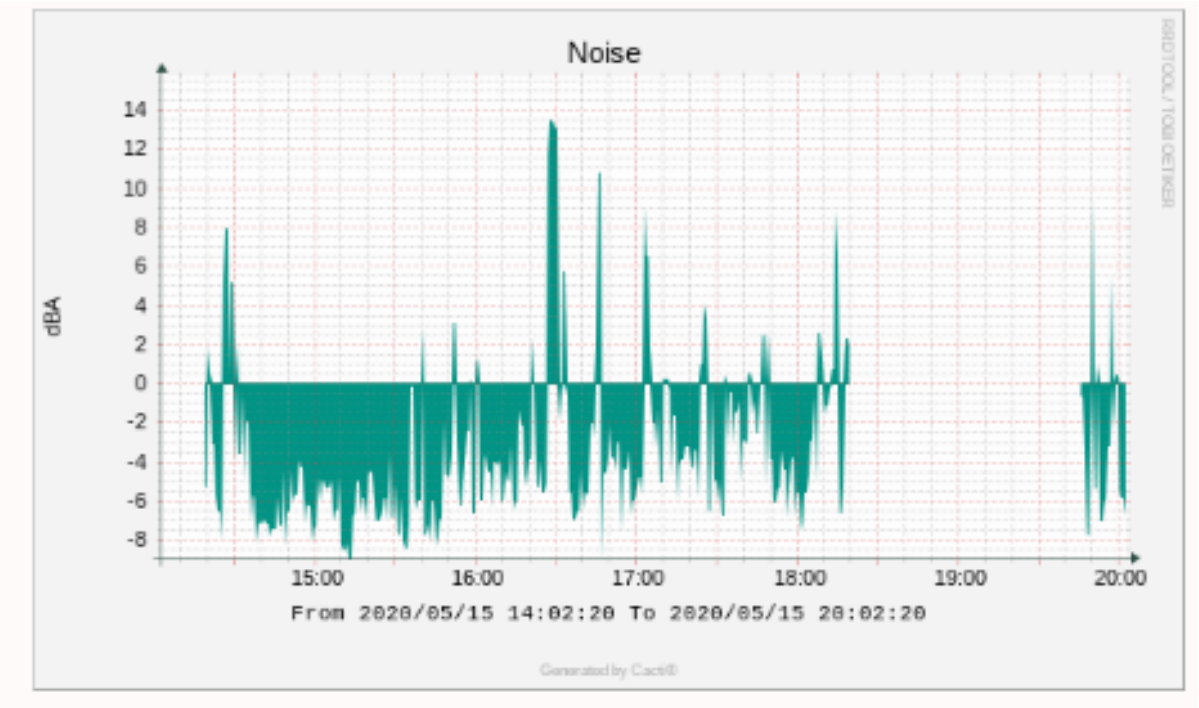
Obrázek 30: Vyťaženie pamäte v časovom intervale dvoch dní.



Obrázek 31: Vyťaženie pamäte v časovom intervale dvoch dní.



Obrázek 32: Svietivosť.



Obrázek 33: Úroveň hluku.

7 Záver

Cieľom diplomovej práce bolo vytvoriť automatizovaný zber zo senzorov mobilného telefónu a následne dáta vykresliť v monitorovacom systéme Cacti.

Odosielanie dát bolo realizované pomocou aplikácie Sensor Node, ktorá odosielala dáta v reálnom čase na mikropočítač Raspberry Pi. Celý zber fungoval pomocou protokolu MQTT, pričom server reprezentoval mikropočítač Rasperry Pi 4 a klienta, ktorý odosielal dáta, reprezentoval mobilný telefón Xiaomi M3.

V teoretickej časti práce som sa venovala problematike senzorov, zberu a spracovania dát zo senzorov pomocou protokolu MQTT. Súčasne teoretická časť práce obsahovala problematiku tvorby časových grafov pomocou RRDtool databázy.

Dáta zo senzorov mobilného telefónu boli odosielané v reálnom čase na brokera, ktorý reprezentoval mikropočítač, a následne sa ukladali do databázy RRDtool. Výsledné grafy zobrazujú ukážky vykreselnia pomocou nástroja Cacti, kde vykreslené dáta zobrazujú využitie pamäte, mieru zaťaženia systému, svietivosť a hluk zo senzorov mobilného telefónu v rôznych časových intervaloch.

Odkazy

[1] S.Ali, S.Khusro, A.Rauf and S.Mahfoo, Saeed. *Sensors and Mobile Phones: Evolution and State-of-the-Art.* [online]. [cit.15.05.2020]. Dostupné z: https://www.researchgate.net/publication/272482886-Sensors-and-Mobile-Phones-Evolution-and-State-of-the-Art

[2] Mohammad Masoud,Yousef Jaradat,Ahmad Manasrah,Ismael Jannoud. *Sensors of Smart Devices in the Internet of Everything (IoE) Era: Big Opportunities and Massive Doubts.* [online]. [cit. 15.05.2020] Dostupné z: https://www.hindawi.com/journals/js/2019/6514520/

[3] Pallavi Sethi and Smruti R. Sarangi. *Internet of Things: Architectures, Protocols, and Applications.* [online]. [cit. 15.05.2020] Dostupné z: https://www.hindawi.com/journals/js/2019/6514520/

[4] IoT Standards and Protocols. NetworkLessons.com | Networking in Plain English [online] . [cit. 15.05.2020] dostupné z: https://networklessons.com/cisco/evolving-technologies/iot-standards-and-protocols.

[5] Sensors Overview | Android Developers. *Android Developers* [online]. [cit. 15.05.2020]. Dostupné z: https://developer.android.com/guide/topics/sensors/sensors-overview.

[6] Introducing the MQTT Protocol - MQTT Essentials: Part 1. HiveMQ - Enterprise ready MQTT to move your IoT data [online] *Hivemq* . [cit. 15.05.2020] Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/.

[7] Publish& Subscribe - MQTT Essentials: Part 2. HiveMQ - Enterprise ready MQTT to move your IoT data [online]. *Hivemq* [cit. 15.05.2020] Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/

[8] MQTT Publish, Subscribe& Unsubscribe - MQTT Essentials: Part 4. HiveMQ - Enterprise ready MQTT to move your IoT data [online]. *Hivemq* [cit. 15.05.2020] Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/

[9] Client, Broker / Server and Connection Establishment - MQTT Essentials: Part 3. HiveMQ - Enterprise ready MQTT to move your IoT data [online] *Hivemq* [cit. 15.05.2020] Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/

[10] Quality of Service 0,1& 2 - MQTT Essentials: Part 6. HiveMQ - Enterprise ready MQTT to move your IoT data [online]. *Hivemq* [cit. 15.05.2020] Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[11] Teach, Learn, and Make with Raspberry Pi – Raspberry Pi. Teach, Learn, and Make with Raspberry Pi – Raspberry Pi [online]. *Teach, Learn, and Make with Raspberry Pi* [cit. 15.05.2020]. Dostupné z: https://www.raspberrypi.org

[12] paho-mqtt · PyPI. PyPI · The Python Package Index [online]. *PyPI. PyPI* [cit. 15.05.2020] Dostupné z: https://pypi.org/project/paho-mqtt/

[13] Sensor Node Free - Apps on Google Play. [online]. Copyright ©2020 Google [cit. 15.05.2020]

[14] Dalibor Zegzulka. *Monitorování a správa síti pomocí nástroje RRDtool.* DSpace VŠB-TUO [online]. [cit. 15.05.2020] Dostupné z: https://https://dspace.vsb.cz/handle/10084/98698